## DOCTORAL THESIS

# 3D SPACE Special Project in Advanced Computer Environments

Patterson, Dale

*Award date:*
2004

# 3D SPACE

*Special Project in Advanced Computer Environments*

**by Dale Andrew Keith Patterson BCompSci (Hons)**

*PhD thesis submitted for the degree of Doctor of Philosophy, in the School of Information Technology, Bond University, December 2003.*

# *Abstract*

The primary objective of this research is to use the benefits offered by computerized **three dimensional graphics** and apply those to the field of **human computer interaction**. Focussing primarily on the interactive content of the 3D world, this research describes a range of innovative new interface elements demonstrating specific new 3D interfaces/components designed to provide a new **interactive 3D** method for handling a range of particular common real-world tasks (ranging from simple value setting tasks up to larger scale systems for browsing structured sets of hierarchical data). These systems incorporate new design concepts such as **active 3D interfaces** that present their data to the user rather than statically waiting for the user to interact with them (these systems prove particularly useful in the presentation of large sets of data). Overall this set of components introduces a range of new interface styles that prove very effective in many mainstream real world tasks.

In addition to the development of these systems, this project demonstrates a new high level 3D interface development tool designed to simplify the challenge of constructing interactive 3D user interfaces and in doing so make **3D interface development** available to a wider developer base. By constructing the components mentioned above in a structured generic form, this combination of a new development tool and a range of re-usable components provides a **strong development platform,** from which more complex **interactive 3D interfaces** can be constructed.

In essence the core idea that underlies this research is making the construction of interactive and functional 3D interfaces simpler to undertake (by developing effective re-usable components to handle mainstream tasks) while at the same time generating resulting 3D interfaces that are more effective and more capable of providing users with an enjoyable and functional 3D working environment.

# *Acknowledgements*

The author would like to acknowledge the contributions of many people to this research including the project supervisors at Bond University, Dr Zheng da Wu and Dr Michael Rees. The support and flexibility they provided throughout the research enabled the project to reach its full potential.

Personal thanks particularly go to Dr. Rees for his inputs in the areas of user testing and interface research and to Dr. Wu for introducing the author to 3D graphics many years ago and providing support in this area throughout the research project.

The author would also like to thank Bond University for its continual support of this research and provision of scholarships, without which this project could not have been completed. Many thanks are also passed on to all of the test users, all of whom freely contributed their time and without whom this research would not exist. Last but not least the author would like to thank his family whose support throughout this project, particularly at the hardest points, helped carry it through to the completion.

# *Contents*

# *Figures*

# *Chapter 1: Introduction*

## 1.1 Motivation

The motivation for this work began with the author undertaking the undergraduate computer graphics course as part of a computer science degree in the early nineties. The ability to develop 3D graphics applications, each of which could take the flat planar screen and turn it into an interactive three dimensional space captured the imagination.

The first project implemented by the author in this area was a simple interactive lamp (modelled on the Luxo Jnr lamp and built using C++ and the PHIGS graphics library [vanDam 88][Foley 90]). After building the basic geometry of the lamp some simple interaction was added to make it possible to grab and drag the lamps head around. Although this was a very simple system, it was immediately obvious that this type of interactive three dimensional software had enormous untapped potential.

Inspired by this untapped potential the author developed several more 3D projects, with each new project opening up new areas of 3D graphics' potential. The world around us is three dimensional, so surely a 3D computer application could tap some of those natural three dimensional skills. Unfortunately, at that time, high end 3D graphics required special hardware and was not readily available to the mainstream public. However with time (and the boom in 3D games) the 3D performance of the mainstream home computer has become powerful enough to support the kind of interactive graphical applications that this research describes. As a result it now seems that the world is ready and waiting for the potential of interactive 3D graphics to be used more effectively. The key motivation for this research is to demonstrate the functionality and usefulness of three dimensional graphics for real world mainstream user interactions.

## 1.2 The Idea

The core idea behind this research is to take 3D graphics applications and apply them to the field of human computer interfaces. This research has a particular focus on the content and interactivity of the items that make up the 3D world or environment.

Coming from a programming/engineering perspective it was clear that the development of a set of reusable "3D components", each designed to provide specific interaction capabilities, would enable the development of complex 3D interfaces through the use of small building block type pieces. Combining a set of such reusable components with an easy to use "construction tool", designed to bring them together into an overall functioning interface, could enable the rapid development of effective and functional 3D interfaces.

In essence the idea that underlies this research is making the development of functional 3D interfaces simpler to undertake whilst at the same time generating more effective results in the final interactive system.

## 1.3 The Contributions

This research contributes to the field of 3D user interfaces in a range of areas including through the demonstration of the effectiveness of high level 3D interface development tools (and associated reusable interface components) for the construction of interactive 3D interfaces. In doing so the project incorporates the development of an easy to use (i.e. high level) *3D interface development* tool. This tool in combination with a set of innovative new 3D components provides its users with a development platform from which they can rapidly create interactive 3D interfaces without needing to undertake the complex 3D graphics programming tasks with which such interfaces are generally related. This tool primarily contributes to the field by enabling a new group of developers, less skilled but very numerous, to enter the field of 3D interface developers through the availability of a higher level 3D interface development tool.

In addition to the development tool itself this research also identifies the need for a set of mainstream 3D interface components. These components represent small self contained pieces designed to provide specific functionality within a broader 3D interface. One of the key contributions made by this research is in identifying, designing, implementing and performance testing a range of new and innovative 3D components. The overall objective being the creation of a core set of reusable elements from which more complex interfaces can be constructed. These components contribute in several areas including:

- Identifying common interaction tasks and demonstrating new mechanisms in 3D that can be applied to those tasks. The tasks specifically addressed include:
    - Selection from large sets of unstructured data (eg. search results) using this projects newly created 3D flow and tunnel components.
    - Selection from large sets of structured data (eg. hierarchical systems, libraries) using this project's newly created 3D circulatory system components.
    - Selection from small sets of data (eg. as menus in a 2D GUI system would be implemented) using this projects newly created 3D drum component.
    - Setting of ranging values (eg. floating point values in fixed ranges like the 2D slider would set in a GUI) using this projects 3D slider component(s).
    - Enabling "mobile users" to carry tools and items with them in a 3D interface using this projects newly created tool-belt, Swiss army knife and pouch components.
- Providing developers with reusable complete systems and sub systems (i.e. components) to handle those common tasks.
- Providing users with a development platform from which to piece together full interfaces from component pieces.

The final and perhaps most broadly interesting contribution of this research is in the form of discoveries made through this development and testing process (eg. new 3D design principles, including a better understanding of user capabilities and preferences).

## 1.4 Evaluation

Evaluation of this research falls into two categories, firstly evaluating how effective the development tool is in providing an effective platform from which interface developers can construct 3D interfaces and secondly how effective each of the individual 3D components is in achieving its particular interactive task.

In evaluating each of the individual 3D components, this research undertakes a series of interactive trials with real world users to determine how effective (or otherwise) each component is, both in terms of numerical/quantitative performance measures and qualitative user opinions of the suitability of the interface to the task.

Assessing the effectiveness of the development tool/environment is also implemented through user trials (i.e. where users construct an interactive interface); however due to a lack of comparative systems (i.e. there are few other 3D interface development tools) the results from these trials are primarily analysed in terms of the users qualitative measures of system effectiveness. For more detail on the assessment of this project see *Chapter 7: Evaluating 3D SPACE - Testing on Users* later in this thesis.

## 1.5 Discussion

The user testing process is one of the more challenging elements of this research, not in terms of the task itself, but in terms of gaining useful results which can be identified as applying in the broader user population. The key issues relating to the effective analysis of trial performance measures fall into several areas:

- **Survey Size & User Variability** - this research uses relatively small survey groups (consisting of ~10 participants per trial). In addition the variability of user results (which is common in this type of user testing) can be a problematic factor. Given the nature of the trials and the associated group sizes this can (in particular cases) be a limiting factor in determining the comparative effectiveness of various systems. As a general principle the results are published using a combination of average results and the “95% confidence intervals” associated with those results. These confidence intervals provide the viewer with a sense of how meaningful any particular piece of data is. That is the tighter the range into which 95% fall the more meaningful that average result is.
- **Bias due to User Experience** - for example comparing an experienced computer users performance when selecting items from a menu against selecting items from a “new component” will generate biased results towards the menu due to the users experience with that interface mechanism. The trials in this research attempt to deal with this bias in two ways:
  - At the start of all trials each user answers a questionnaire to indicate his/her previous experience with both computers in general and 3D systems. By recording this information we can analyze the result data based on experience levels.
  - All trials are implemented as 3D interfaces; in cases where the trial system is being compared to an essentially 2D system (e.g. a menu) this menu is

implemented in its two dimensional form within the 3D environment. Therefore it is a 2D menu in a 3D environment; this does not totally remove the experience factor but it does provide an equal base platform (i.e. the 3D environment) in which both systems are tested.

Given these factors the results from the trials are generally intended to demonstrate the potential of the various systems rather than prove their comparative effectiveness (although some systems do achieve this). In publishing the results the variability of user performance is accounted for by making use of the statistical concept of the "95% confidence interval". This concept is absolutely correct when applied to normal distributions, which is the type of distribution that would be expected from user testing of this type. However it is important to note the fact that this research makes the assumption that user testing results will fall into a normal distribution and hence the "95% confidence interval" will apply.

## 1.6 Organization

This textual description of the interactive systems described in this research could never match the experience of actually interacting with these systems and as such the author hopes that the descriptions provided in this document effectively convey the workings of these systems. The author would love to be able to incorporate interactive content (i.e. animations and the interactive components themselves) in this thesis, however the textual format limits this. Clearly electronic forms of document, where this interactive content could be embedded represent the future for presenting visual/interactive research of this kind and it is hoped that such systems become more widely used for presentation of research projects of this kind in the future. The following points describe the layout of this document, including brief descriptions of the content of the various chapters.

The *Introduction* chapter (i.e. the current chapter) broadly introduces this project, the motivation for it, its topic, how it has been assessed and touches briefly on what the research has achieved.

The thesis sections that follow begin with the *Background* chapter. This chapter looks in some detail at the previous research that as been undertaken in the field of 3D user interfaces. Looking at the innovative systems developed in both interface applications, tools for constructing such applications as well as general design principles in 3D interfaces. The subsequent chapter *Challenges for 3D Interfaces* then analyses the state of the 3D interface field and outlines the issues that need to be addressed to develop this field further. Moving on from the *Challenges for 3D Interfaces* is the *Overview of the 3D SPACE Project* section, which looks at this research project and where it fits in terms of its goals, the methodology used in achieving those goals and introduces the broad conceptual ideas that are the basis of this research, which is titled the "3D SPACE" project.

From that point we move away from the conceptual issues and look at the practical side of the research, in particular *The 3D Interface Construction Tool* chapter deals with the design and implementation of the 3D interface builder application. This application is a development tool constructed to enable the rapid development of interactive 3D user interfaces. In this chapter the tool is described in terms of its features, how it is

implemented and how it functions in its task. It is in these, implementation related sections that you will notice code examples in the form shown below:

```
// Code Example

cout << “Hello World!” << flush;          // write Hello World to screen
…
```

This representation of code, using double slash green highlighting for comments is broadly maintained throughout the thesis. Code examples will be in one of either C++, Java or JavaScript, where relevant an indication will be given for which language is being used.

Following the *3D Interface Construction Tool* chapter the next chapter *The Set of 3D Interface Components* contains a series of subsections each of which describes a single new 3D interface component. Each of these components is designed to deal with a particular real-world interaction task from within a 3D interface. In these sections each component is described in terms of its design, how it functions and how it is implemented. In the following chapter, *Evaluating 3D SPACE*, each of these components is tested and the testing methods and results of these performance tests is outlined. In addition to testing the components, this section also includes evaluations and results for user testing with the builder application itself.

The final descriptive chapter of the thesis, *Conclusions and Contributions*, summarizes the results and outlines the conclusions that can be made from those results. It also looks at the overall contributions that this research has made and how those contributions apply in real world tasks.

# *Chapter 2: Background*

Although the field of 3D user interfaces is a relatively recent development, it is a field that is founded on a base of underlying technologies (such as hardware technologies, design principles etc), without which it could not exist. During its short lifetime a range of innovative designs and interfaces have been developed. This section takes a detailed look at those systems as well as looking at a collection of other relevant related research (such as development tools). This background provides a strong base from which to understand where this research project fits into the grander scheme of 3D interface research.

The remainder of this chapter summarizes both the previous work and current state of the art in this field. This summary is divided into sections based on the general topics involved in 3D user interface research, including:

- **3D Interfaces - The Elements Involved.**

  A brief description of the elements that make up the field of 3D Human-Computer Interfaces.

- **3D Interface Principles.**

  Looks at what is already known regarding design principles for 3D interfaces. This section includes human-factors studies as well as other principles for the creation of effective 3D interfaces.

- **3D Interface Applications.**

  Covers the existing range of 3D interfaces and their achievements.

- **3D Interface Development Tools.**

  Looks at existing attempts to build libraries and tools to support the development of 3D Interfaces.

- **Summary.**

  Briefly combines the information from the above sections into a summary of where the field of 3D Human-Computer Interfaces stands today.

## 2.1 3D Interfaces - The Elements Involved

Since the start of the computer age people have been attempting to improve the computer's ability to relate to (or interface with) the human user. These efforts have steadily succeeded in developing more effective Human-Computer Interfaces (HCI). The improvements in HCI have always been closely linked to developments in underlying technologies (e.g. without graphical displays, graphics systems & software the 2D Graphical User Interface would never have developed). This reliance on underlying technologies is particularly important to the field of 3D Human-Computer Interfaces (3D User Interfaces). In essence 3D interfaces cannot exist without the underlying support of a range of other technologies, many of which have only developed in recent times. Each of these underlying technologies has a direct influence on specific elements within the 3D

Human-Computer interface. This PhD project is particularly interested in the software side of the 3D interface, however it is still important to briefly look at the hardware issues faced by 3D User Interfaces.

A human-computer interface is actually made up of a series of levels of abstraction between the human user and computer. The first level of abstraction is in the form of interaction devices. These devices are the only means for the user to receive information from the computer and pass information back down to the computer. The area of three-dimensional device design is one of the most active research areas. This is primarily due to the fact that the field of 3D interfaces is so new and the development community is yet to standardize on the most suitable devices for use in three-dimensional space.

From the user's perspective the next level of abstraction is the interface application itself (i.e. the software that controls how the interface looks and reacts). However from an interface development perspective there are several additional levels of abstraction between interaction devices and the interface application itself, those levels include the following:

- **System Hardware**: the physical hardware of the machine. The underlying operating system software controls the actions performed by this hardware.
- **Operating System Software**: the software that handles the underlying hardware and manages the software activities. It includes the following sub-features that are relevant to 3D interfaces.
  - *Device Driving Software***:** the software that controls the flow of information to/from the interaction devices.
  - *Hardware/Software Interface*: "Application Programming Interface" to hardware facilities, the software that allows other applications access to hardware capabilities
- **Programming Languages, Libraries**: these software applications allow users to create applications of their own, by using the facilities provided by the Operating System and the programming language
  - *3D Graphics Libraries*: these are libraries that provide the programming capable user access to the 3D graphics rendering capabilities of the Operating System.
- **3D Interface Libraries**: this software makes use of functions provided by the Operating System and other libraries to provide a simpler method (in a programming sense an Application Programming Interface) for the development of 3D User interfaces by those with programming skills.
  - These libraries are optional as the developer can create a 3D interface by directly using the functionality provided by the Operating System.
- **3D Interface Application Software**: this is the software that controls the look and feel of the actual interface. It achieves this in one of two ways:
  - By making use of facilities provided by the above mentioned Graphics Libraries, Interface Libraries and Operating System functionality; to create a stand-alone interface application

- Or by using a Viewer application (which uses Operating System etc. support) and an interface/scene description language**.** This description of the interface can then be used by the 'viewer' application to run or view the interface. With VRML (i.e. Virtual Reality Modeling Language), (or the newer X3D) as its primary example, this new 'portable' level of abstraction has developed as a popular method of interface description.

The 3D SPACE research project is particularly interested in the last two levels of interface abstraction, '*3D Interface Libraries*' and '*3D Interface Application Software*', as a result the bulk of the following research focuses on those topics.

Despite the focus on those topics the following sections will give a brief history as well as a summary of current research in all of the fields mentioned in relation to 3D Human-Computer interfaces.

## 2.2 3D Interface Principles

Understanding the user and his capabilities is a critical step in any attempt to produce more effective Human-Computer Interfaces. Interface principles are really a set of heuristics regarding how to best present a given type of information to the user. In two-dimensional graphical user interfaces these principles are well developed, unfortunately in three-dimensional interfaces such principles are limited. However it is true that many principles from two-dimensional interfaces will still apply in 3D interfaces.

Despite the fact that 3D interfaces have only existed for a short time, user studies and simple trial and error have already established many of the principles for developing effective 3D interfaces. General principles that broadly describe how a three-dimensional interface should operate are currently rarely recorded in 'principle' form. These principles are largely incorporated either intentionally or unintentionally within the range of 3D interface applications.

Despite the fact that 3D Interaction Principles are still in their early stages, general principles (those that exist) relating to the software side of the 3D interface, can be summarized as follows:

- **Presence Principles**: giving the user a presence in the virtual environment is critical to interface success (techniques include [Fisher 89] '*virtual hand*', [Wloka 95] '*tool metaphor'*, [Hinkley 97] '*two-hand*' and '*grounding table*').
- **Navigation Principles**: giving the user effective mechanisms to move around and navigate in three-dimensional space (see following section for more details).
- **Interaction Principles**: enabling the user to interact with the items in his/her environment via user interface mechanisms (see following section).
- **Presentation Principles**: principles regarding 'how' three-dimensional information should be presented to the user ([Robertson 91], [Eisenberg 97]).
- **Multi-user Principles**: principles and issues arising in multi-user environments.
- **General Usability Principles**: principles regarding the control of 'general' environmental factors.

### 2.2.1 Presentation Principles

Early research [Robertson 91] established that information visualization using 3D graphics (an interface) can be used to stimulate recognition of patterns and structure in information. It does so by exploiting the human perceptual system. This established the fact that humans can perceive the structure and relationships between three-dimensionally presented information in a three-dimensional space. This is an important principle because it shows that the actual physical structure of a 3D interface can influence its effectiveness in communicating information.

In 1997 the work of Eisenberg extended this principle by studying how humans interpret three-dimensional shapes (i.e. polyhedra) [Eisenberg 97]. Eisenberg's research found that humans perceive a 12 sided dodecahedron (spherical shape) to be more complex than a 12 sided decagonal prism (twelve sided cylinder). Further research uncovered some fundamental concepts of how humans interpret three-dimensional shapes, and which features make a shape 'easier' to understand and work with.

Desirable features in the presentation of a shape are:

- A preponderance of non-diagonal edges (with a preference for vertical over horizontal)
- A vertical axis of rotational symmetry
- Bilateral Symmetry
- Stability (i.e. the apparent center of gravity is over either a flat bottom face or is directly over the bottom most point of the shape)

[Eisenberg 97] concludes that people visualize shapes depending on their orientation and hence presenting them in the best orientation can make them easier to work with.

### 2.2.2 Presence Principles

A popular 'presence' oriented principle involves placing 'virtual' equivalents of real-world interaction devices into the 3D interface. Starting with the early VR research [Sutherland 65, 68] at centres like NASA Ames [Fisher 86, 89], virtual equivalents to real-world interaction devices have been widely identified as critical features in an effective 3D interface (e.g. a virtual hand to match the users real hand/glove device). This is one of the only principles that has been broadly adopted by all three-dimensional interfaces. As a principle it offers several advantages to the interface including:

- **Representation of Interaction Device**: the virtual equivalent is easily recognized and hence its functions are more clearly understood (due to user's real-world experience, commonly referred to as the 'real-world' metaphor) [Bowman 97].
- **Reference Point**: the virtual device gives the user a reference point in the virtual world (separate to the viewpoint). This reference point provides cues regarding the orientation (and other relationships) of the virtual world relative to the real world.

The fundamental concept of relating real and virtual devices has been extended in various ways including [Wloka 95] the 'tool metaphor', which extends the simple representation

of a hand or location into a virtual tool. The concept here is that where the ‘virtual hand’ concept can provide presence and reference points, the ‘virtual tool’ can also provide specialist interactions with the world, much as people use tools to achieve specific tasks in real life.

Another extension of this concept is that of ‘two-handed’ interaction [Bier 93][Hinkley 97]. Early two-handed systems such as [Bier 93]'s ‘*Toolglass and Magic Lenses*’ allowed two-handed interaction with a two-dimensional interface. Later research [Hinkley 97] extends this two-dimensional principle into three-dimensional interfaces. Hinkley’s dual-hand research has shown that using both hands can help users gain a better sense of the three-dimensional space they are working in. Another option for providing the benefits provided by two-hands is the concept of a grounding table (for the hand(s) to work/push on), this device would provide a generic ‘working space’ (in both real and virtual) for the user to interact.

### *2.2.3 Navigation Principles*

The spatial nature of three-dimensional interfaces introduces the problem of navigation through that space. Early research found that users were easily confused when attempting to navigate in 3D space.

> ‘ Virtual world navigators may wander aimlessly when attempting to find a place for the first time. They may have difficulty relocating places recently visited. They are often unable to grasp the overall topological structure of the space ' [Darken 95]

To overcome this problem the three-dimensional interface needs to implement ‘navigation aids’ to assist the user.

In 1993 Rudy Darken developed ‘A Toolset for Navigation in Virtual Environments’, this toolset consisted of a collection of navigation techniques modelled on real-world navigation methods [Darken 93]. The methods implemented included flying, spatial audio, breadcrumb markers (allowing user to leave markers as he goes; to follow later), coordinate feedback, districting (dividing the environment into a collection of large sections), landmarks, grid navigation and mapview. The principles of human spatial navigation they discovered were:

- People tend to take advantage of environmental cues in predictable ways (partitioning spaces, maintaining relationships).
- Cues in different modalities (visual & audio) make targets easier to find.
- The tools that are available influence the way users navigate through the space.
- Cartographic design principles can be extended from 2D to 3D space (for planar navigation at least)

Several years later Darken used the principles discovered above to develop a set of principles to be applied to an interface to improve user navigation [Darken 95]. Those principles were:

- Divide a large scale world into distinct small parts, preserving a sense of place

- Organize small parts under a simple organizational principle (such as street design)
- Provide frequent directional queues

Part of this research also outlines a set of principles to improve map design for wayfinding tasks:

- Show all organizational elements (paths, landmarks, districts, etc.) and the organizational principle
- Always show the observer's position
- Orient the map with respect to the observer so that the forward-up equivalence principle is accommodated [Darken 95].

At the same time (1995) a new navigation concept was being developed called *'Worlds In Miniature'* (WIM) [Pausch 95]. A WIM is a hand-held miniature graphical representation of the virtual environment, similar to a map cube. When the user moves an object in the WIM, that object moves in the real virtual environment (VE). When he moves the representation of himself[1] in the WIM, he moves/flies in the real VE. In essence the WIM concept is based on a 3D map, however Pausch extends that concept in several ways. Firstly they use a delayed update on moves of the representation of the user (i.e. when he moves the representation of himself the 'real VE' is not updated until he stops (completes) the move (this reduces the confusion caused by constant update and keeps his focus in the WIM). Finding that users focused entirely on the WIM during moves they decided to make the WIM fill the screen during the move and then have it fade out to be replaced by a new small WIM for the new location. For example the user moves the doll, the WIM expands and the doll is seen to move in the old WIMs' space, old WIM fades out and the new view is shown (with a new WIM). Users found this technique more effective.

The second feature, that is an extension over a basic map, is the concept of using multiple WIMS for differing parts of the system.

> ' With multiple WIMs, each WIM acts as a portal onto a different, perhaps distant, part of the surrounding immersive world, or to a different world all together.' [Pausch 95].

By moving the doll icon from one WIM to another, the user can quickly transport his immersive viewpoint to another point in space, or to a completely different context.

> ' Consider another case, where an external entity (e.g. the system or another user), aggressively forces a change in the users viewpoint. The new location can first appear as a WIM in the current location, providing constancy for the forced navigation' [Pausch 95].

[Stoakey 95] and [Elvins 97] '*Worldlets*', are also based on the '*World In Miniature*' concept. Elvins identifies that the WIM concept is less effective in non-enclosed space

---

[1] Users preferred a 'doll' to a camera to represent them in the WIM.

(e.g. in an open starfield the WIM is functionally no better than the main view); also the WIM itself obstructs the user's view of the world. Elvins introduces a general-purpose technique for creating 3D landmark representations. It does not attempt to allow users to interact with the 'worldlet' it is merely a 3D version of a landmark or portal, primarily to replace text or images in a viewpoint set. Although this represents a step back in terms of interactivity from the WIM concept, the use of 'worldlets' for landmark representation removes the interactive features of the WIM and in doing so isolates one valuable feature of the WIM concept. The WIM/Worldlets concept has also been effectively applied in augmented reality systems [Bell 02] using Java3D. [Haik 02] uses Web3D (i.e. VRML) to construct and assess navigational performance with a WIM based navigational aid.

Viewing interaction in terms of the users perspective enabled [Mohageg 96] whilst designing and constructing of the "WebSpace VRML browser" application to identify the following design issues relating to implementing user navigation through the virtual space (i.e. as this is a responsibility of the VRML browser application).

> 'While participants could navigate to desired locations, they often commented on not being completely sure of their current location in relation to other locales/landmarks in the scene.
>
> The most interesting discovery was that users were not very motivated to employ UI fixtures to navigate a scene. Users expected to simply click in the viewing window and have some sort of navigation take place automatically. Note that this is in direct conflict with our intent to be consistent with web conventions by reserving clicking in the viewing window for traversing links. It was very clear, however, that many novices expected to click on an object or area of interest and be taken there. They seemed to want to use the cursor as a pointer that would whisk them to their location of interest within the scene. This is the functionality we had defined for the Seek Tool, but many users wanted this behavior to be a primary navigation mechanism. Our challenge now was to support this seeking behavior without violating the link following convention.' [Mohageg 96].

Essentially this identified that most 3D interface users wanted to stay contained within the 3D space. In general they wanted to be able to obtain state information (eg. bird's eye view as described in the WIM systems) and control their interactions and navigation from within the 3D space rather than using a separate interface tool.

> ' Many (users) anticipated being able to click on an in-scene object to take them to places of interest. The viewpoints menu contains points of interest; however, users prefer to maintain focus on the content rather than access a menu outside the main viewing window.' [Mohageg 96].

The issue of scale is an important factor in three-dimensional navigation (e.g. the starfield example shows how an effective technique at one scale can be ineffective at a larger scale). [Ware 97] describes a method for controlling flying speed in a VR flying interface, this method attempts to deal with scale by altering the flying speed as needed to reflect the scale of the world being flown through. The 'point of view' method allows the user to fly between pre-defined views of a scene (the pace of flying is defined relative to the distance between the pre-defined views). This is an effective method but it requires information about the destination point before starting. The concept proposed to obtain

similar effectiveness without needing a pre-defined end point, by working out the distance held in the Z buffer and from that determining approximate speeds (it also uses information on the distribution of objects within the depth buffers space to improve speeds). This method of determining scale is very basic but is much more effective than fixed speeds, and in some way automatically handles the issue of varying scales.

> ' Overall our impression is that modulating velocity based on depth sampling creates a markedly improved exploration interface, especially where large changes of scale are involved ' [Ware 97].

As the research indicates there are numerous principles being applied to three-dimensional navigation, none of which have been accepted as standards. The lack of navigation standards is seen as a major limitation of 3D interfaces.

> ‘ We desperately need a set of 'standard interface cues’ for navigating in virtual space’ [Damer 96]

### *2.2.4 Interaction Principles*

Principles about how users can interact with elements in the 3D interface tend to vary based on the system being viewed, however there are some common underlying principles including:

#### Direct Interaction

Direct Interaction refers to interacting with objects in the viewers local region, that is, interacting with objects that are in the users hand (or equivalent), this is also referred to as the ‘real-world’ metaphor (i.e. like the real world the user can only interact with objects that are within arms reach). The user can more effectively interact with an object if that object is in close proximity (e.g. in the users hands) [Mine 97]. With this in mind most interaction principles relate to working with objects in the users hands. According to [Mine 97] working within a users natural working volume has the following advantages:

- Takes advantage of proprioceptive information[2]
- Provides a more direct mapping between hand motion and object motion
- Yields stronger three-dimensional cues (perspective, stereopsis etc. are more pronounced when nearer the viewer)
- Provides finer angular precision of motion

Despite the advantages of interacting with objects only in a local sense, in a functional 3D interface the user often needs to interact with objects at long range. Interestingly most of the long-range interaction principles involve either bringing the remote object to the user or moving the user to the object.

---

[2] Proprioception: a person’s sense of the position and orientation of his body and limbs.

## Interaction at a Distance

Three dimensional interface developers face a major challenge in attempting to allow users to interact with objects that are outside their local interaction range. Early techniques such as 'Go-Go arms', arms that simply extend to the object as needed, were found to be too difficult to work with. Current interaction systems tend to be based on one of two options:

- **Bringing the Object to the User:**
  - [Mine 97] '*Scaled world grab*' – the world is automatically scaled down about the users head every time he grabs an object and scaled back when he releases it (the scale factor is based on the object distance).
  - [Pierce 97a] outlines interacting on the image plane (i.e. interacting with the geometry as projected onto the viewing plane) rather than interacting with the 3D geometry. Although not a true immersive 3D technique this essentially involves using the projection to bring the object to the user.
  - [Pierce 99a] *'Voodoo Dolls'* - describes a system of using an abstracted version of an item in the world (i.e. the voodoo doll) and allowing the user to manipulate the local version and have it take effect on the distant item.
- **Taking the User to the Object:** examples include:
  - [Bowman 97] '*HOMER*' (Hand-centered Object Manipulation Extending Ray Casting) – Firstly the user selects the object using a ray-casting technique (i.e. line through space) the virtual hand then moves in space to the grabbed object (on completion the hand moves back to the normal position). Object manipulation is then undertaken relative to the users body (as though the hand were in the local space). The advantages include:
    - Object grabbing is easier (i.e. uses most effective RAY technique)
    - Objects at any distance can be selected with no additional/tiring arm actions
    - Object manipulation requires less physical effort
    - Object can be manipulated at any distance away from users position
  - [Pausch 95] '*Worlds In Miniature*'[3]- allows interaction with objects in the miniature world which is always located locally (i.e. in arms reach)

Interestingly there have been few attempts to create an interaction system which functions at a great distance. In general it is broadly accepted that users need to interact within their natural working volume. The research of [Feiner 93] and [Mine 97] both demonstrate the user desire to work in a body relative coordinate system rather than object or world based coordinate systems.

[3] World In Miniature is discussed in the earlier section 'Navigation Principles'.

One of the interesting principles established by [Bowman 97] is the fact that 'grabbing' and 'interaction' should be treated as completely separate tasks (this is reflected in their HOMER system using ray-casting for selection and direct manipulation for interaction).

## See-through Interfaces

Almost all three-dimensional presentation of information relies on depth cues identified in psychological research on human perception in the natural environment. The most commonly exploited depth cues include occlusion, perspective, shadows, texture, binocular disparity, motion parallax and active movement.

Research into 'see-trough' interfaces proposes that partial-occlusion caused by semi-transparent objects can provide excellent depth cues without totally blocking the view.

In 1993 [Bier 93] developed the concept of a two-dimensionally based 'see-through' interface, 'Toolglass and Magic Lenses'. This research identified many of the benefits that 'partially transparent' interfaces can supply in a two-dimensional interface:

- ***Partial transparency enables more effective use of screen space*:** That is, the partially transparent tool can exist in the same screen space as the underlying application (allowing the user to use both in parallel)
- ***Effective means of presenting 'global' or 'environmental' information*:** This information can be displayed as part of the see-through interface without needing to interrupt the underlying application
- ***Enables users direct feedback from actions occurring beneath the cursor*:** That is, the cursor does not obstruct the users view of the underlying task.

The 'see-through' interface was extended into three-dimensional space by [Viega 96] in the form of '*3D Magic Lenses*' and [Zhai 94] with the '*Silk Cursor*'.

The '*Silk Cursor*' interface introduces two primary concepts, firstly the 'tracking symbol' (like the pointer in a GUI) is a 3D volume rather than a point. Secondly (this is the main focus of the research) the surface of the tracking volume[4] is semi-transparent (allowing vision through the tracking volume). The tracking volume is like a clear box covered by a silk stocking, this partial transparency gives the user some additional depth information (i.e. Objects behind the track volume are blocked partially by 2 levels of silk, objects inside only 1 level and objects in front are not blocked at all). This additional depth information is a massive aid to the user in determining how the acquisition volume and the target relate in three-dimensional space. When tested with both stereo and mono viewing systems it was found to be more effective than either a wireframe or solid tracking volume (for both stereo and mono). This research clearly shows the advantages of using a semi-transparent acquisition device for the task of three-dimensional target acquisition.

The '*3D Magic Lenses*' interface extended this research in another way, by identifying the advantages of using partial transparency for interface features other than just the

[4] The tracking volume is the name they use to describe the 3D pointer/locator item in the display.

‘tracking volume’, this concept is also reflected in [Zhai 96]. Between them they identify several existing three-dimensional interfaces which effectively use ‘partial-occlusion’ including ‘*Cone Trees*’ [Robertson 91b], ‘*Worlds In Miniature*’ [Pausch 95], ‘*Surgical Visualization*’ [Hinkley 94] and the ‘*Silk Cursor*’ [Zhai 94].

From this research it is established that many 'widgets' in 3D should be semi-transparent, as they would not interfere with the world if they were. As an example [Zhai 96] demonstrates a semi-transparent hand for VR (an extension of the silk cursor concept).

### Applying Existing 2D Principles

[Wang 99] undertook an interesting study looking into whether the well established Fitts law from 2D systems could be applied in 3D. This study reported:

> ‘ It is interesting to note that results of this study do not fit Fitts’ law. In general, the task completion time did not increase as the target size decreased; this depended on cursor size. Actually, the target size alone showed no significant effects on either the task completion time, transportation time or orientation time. This demonstrates that multidimensional docking or matching tasks are not Fitts’ tasks per se. This further suggests that human information processing for multidimensional object transportation and orientation may be very different from that for pointing.’

Given that the 3D task does not fit the existing 2D method, this research outlines the following conclusions about how 3D systems work:

> ‘We conclude from this study:
>
> 1). Relative sizes of controller, cursor, and target matter in object manipulation.
>
> 2). Same sizes of controller and cursor improve human performance in object manipulation speed.
>
> 3). Same sizes of cursor and target improve human performance in object manipulation accuracy.
>
> 4). Relative size effects should be considered in HCI research and design.’ [Wang 99].

Although some 2D principles can be successfully applied in a 3D interface, it is apparent that the in the areas outlined above 2D principles do not function in a 3D environment.

## *2.2.5 Multi-User Principles*

The possibility of creating ‘virtual’ environments with multiple active users offers many attractive benefits, especially given the recent boom of the Internet. Fortunately many of the principles developed for single user environments still apply effectively to multi-user environments. However the concept of a multi-user environment introduces many new issues which must be dealt with by the user interface. The main issues revolve around how the users will interact and communicate with each other.

In 1993 [Fahlen 93] developed the concept of an ‘aura’. The ‘aura’ method introduces the concept of immersive users having an aura (within which they can communicate with other users/computer generated). The aura is simply an interaction sphere (rendered in

wireframe around the users geometry) which represents the users 'active' communication range.[5] When the aura hits another users aura the two users can agree to open communication channels (e.g. User A's aura hits User B's, they can now talk to each other via a simple text transfer method, other methods are possible including voice, video and others). This Aura principle can also be extended to allow one user to talk to many by taking a 'podium' aura (which allows the speaker to speak and be heard by all other users). The principle can also be used to create small 'conference' groups of users who can all communicate together in a shared way. The underlying principle of allowing a user to control his interaction range in this way is an effective multi-user principle.

Despite the success of the 'aura' concept, the multi-user environment still has many limitations. User studies such as [Damer 96] identify some of the key limitations:

- Confusion relating to when and with whom collisions occur (proposes using auditory cues when collisions occur).
- Distinguishing between users is difficult as avatars[6] are primitive and not unique to people.
- Lack of control over who is speaking at any given time leads to multiple interruptions.

### *2.2.6 General Principles*

In the area of general design issues in 1994 Bryson attempted to document several of the principles that are effective in three-dimensional interfaces [Bryson 94].

- Each object should have its own virtual controllers associated with it.
- Avoid global environment control directly through the hardware interface, that is, avoid gestures that have meaning all the time. It is too easy to make such commands unintentionally.
- Hide virtual controls as much as possible, but make them easy to get at.

[Bryson 94] argues that the user should have access to displays that reflect the state of the environment [Vince 95].

User studies such as [Salzman 96] also provide a selection of general principles, established by studying user performance when interacting with a 3D interface. These principles include:

- Users exhibit noticeable differences in their interaction styles and levels of interaction success.
- Students were shown to learn more effectively when information was presented over multiple short VR experiences rather than a single longer experience

---

[5] The user is able to alter the range of his aura as needed.

[6] Avatars: Computer representations of users.

- Standard approaches to building 2D micro-worlds (GUIs and activities based on planar context) do not scale well to 3D worlds (i.e. 3D worlds need to be designed to make use of 3D space).
- Multimodal interaction (voice, virtual & physical controls) appear to enhance learning (allowing different methods of accessing information gives users a chance to use their own preferred style and hence obtain information more easily).
- Multisensory cues can engage users and direct their attention as needed to avoid problems or focus on issues.
- VR aided students to visualize complex concepts (e.g. chemical bonding)
- Personalizing the VR setup (devices and interface) improves performance significantly (hence flexible ('personalizable') user interface features are a good idea)

The positive use of audio capabilities as identified in [Salzman 96] and shown in systems such as the data mountain is one of several examples where audio outputs/interactions have been effectively applied in 3D interfaces. [Mereu 96] demonstrates the usefulness of audio in aiding performance of all users (in particular visually impaired users) in virtual environments.

> 'The three audio environments map tonal, musical and orchestral sounds to an (x, y, z) position in a 3D environment. In each environment the user's task is to locate a target in three dimensions as accurately and quickly as possible. This experiment has three important results: that audio feedback improves performance in 3D applications for all users; that visually impaired users can use 3D applications with the accuracy of sighted users; and that visually impaired users can attain greater target accuracy than sighted users in a sound-only environment' [Mereu 96].

In the same way that audio cues can aid user performance in 3D [Hubona 99] demonstrates the positive impact that effects such as shadows have on our perception, in particular our perception of depth. Given that shadows are often not used in 3D graphics applications or interfaces the potential for applying shadows to a powerful effect is clear.

Animation, or motion of items within an interface is more widely used and has a base in both two dimensional interface design and also in earlier fundamental animation techniques [Lasseter 87][Johnson 81]. The use of motion and its display to the screen provides a base from which most modern 3D user interfaces are built.

Ideally a complete set of three-dimensional interaction principles would already exist, unfortunately they currently do not.

> ' Virtual Environments lack a unifying framework for interaction, such as the desktop metaphor used in conventional through-the-window computer applications.
>
> The difference between working in a conventional computer environment and working immersed are analogous to the differences between a craftsman at a workbench and one moving about a worksite wearing a toolbelt. His toolbelt had better be large and filled with powerful tools ' [Mine 97].

## 2.3 3D Interface Development Tools

Interface development tools provide developers with a set of mechanisms for implementing interactive 3D systems. These tools come in a range of forms each of which provides a differing level of functionality and accessibility. The earliest forms of development tools offered little more than the underlying OS system or hardware features, however with time tools have become both easier to use and more powerful in terms of what they can implement. Development tools fall into one of a set of categories, ranging from low level graphics programming libraries up to high level world description formats, each of these is described in more detail below.

### *2.3.1 Three Dimensional Graphics Libraries*

Early three-dimensional graphics libraries such as GL[7], PHIGS, GKS and PHIGS+[8] [vanDam 88][Foley 90] provided APIs specifically designed to allow programmers to create three-dimensional graphical applications. Unfortunately due to a combination of limited hardware capabilities and poor cross-platform acceptance, these libraries did not achieve mainstream popularity.

The development and wide cross-platform acceptance of the OpenGL standard [Neider 93], brought the development of three-dimensional graphics programs into a new era. With the ability to develop applications for multiple platforms, OpenGL led to a growth in the development of three-dimensional applications. Despite its popularity OpenGL is still a low-level immediate mode rendering oriented graphics library.

In 1992 Silicon Graphics '*Inventor*' Graphics Toolkit [Strauss 93] extended GL's (later '*OpenInventor*' would be based on OpenGL's) capabilities by further removing the developer from the underlying rendering oriented detail, enabling developers to spend more time working on the content of their three-dimensional environments.

> ' Inventor allows programmers to quickly develop interactive 3D graphics programs of all sorts, based on the concepts of scene structure and object description' [Hartman 96].

With the development of OpenGL and higher level toolkits such as OpenInventor and Java3D, three-dimensional graphics capabilities have become more widely used and as a result three-dimensional user interfaces have become a real possibility for those with the necessary programming skills.

### *2.3.2 VRML a 3D Interface Platform*

From its beginnings at the first annual World Wide Web conference in May 1994, the Virtual Reality Modeling Language[9] has had a significant impact on three-dimensional user interfaces [Hartman 96].

---

[7] GL was Silicon Graphics proprietary 3D graphics library.

[8] PHIGS and GKS were ANSI and ISO standard based graphics libraries.

[9] Originally the M stood for Markup, this was later be changed to Modeling.

The development of the official Virtual Reality Modeling Language (VRML) Version 1.0 provided developers with a new mechanism for constructing and using three-dimensional user interfaces. Previously developers had created stand-alone applications using graphics libraries (e.g. OpenGL), these applications could then be executed on the users machine.

The VRML system was somewhat different; it involved an additional level of abstraction between the three-dimensional interface and the users machine. This abstraction is achieved through the fact that a VRML file is a description of the contents of the virtual environment, it is not an executable application. This descriptive file needs to be loaded into a 'viewer' application (the viewer is an executing application) and then the viewer application can 'display' the virtual environment described by the VRML file.

Version 1.0 of VRML, which was released in October of 1994, was limited to rigid three-dimensional worlds. This limitation made VRML 1.0 of little use to three-dimensional interface developers, as an interface requires some form of interactivity. However in August 1996 a new VRML specification was released (i.e. version 2.0) [Hartman 96], this specification entitled 'Moving Worlds' added many new features including:

- Enhanced Static Worlds (addition of sound, new primitive types e.g. extrusion, backgrounds, animated textures etc)
- Interaction (collision detection capabilities and sensors enable the world to receive input from the user)
- Animation and Behavior Scripting (interpolation to enable keyframe animation and scripting to enable the developer to control interactions and simulations)
- Prototyping new VRML objects (ability to create re-usable VRML components)

VRML 2.0 as described above offered the potential of an interactive 3D space without the need to program all of the details of the world (although some of the interactive features still required programming skills). However despite early popularity, which saw VRML browser/plug-ins added to several mainstream web browser applications, VRML 2.0 failed to gain a stable foothold and eventually lacked the widespread support that was required (eg there was no wide acceptance or usage of the VRML standard (e.g. None of the mainstream WWW[10] browsers (currently) include VRML viewers as a standard feature).

The X3D standard represents the latest version of this VRML family. From the developers point of view X3D offers greater flexibility than the earlier versions, but at this time X3D is still in its early developmental stages. The X3D standard extends VRML to make it more flexible [Web3D 03]. For more detail on the VRML/X3D standards and how these technologies work see the authors summary of this technology [Patterson 03a,b].

VRML (and its variants) and Java3D [Java3D 03] both represent a higher level method of implementing an interactive 3D environment. An object oriented extension to the basic VRML concept is outlined in [Beeson 97].

---

[10] WWW stands for World Wide Web.

In a similar higher level vein, [Dorner 00] demonstrates the concept of "3D Bean Components". The objective of these components is to demonstrate how the component idea (i.e. reusable elements) can be transferred to authoring of 3D content for the WWW. Implemented in Java3D this system, although relatively primitive, identifies the clear advantage that reusable elements can have in simplifying the development task. Essentially an enhanced programming tool this "3D Bean Component" system demonstrates that simple pieces (e.g. the example Beans are very simple shapes) can be written once (in Java/Java3D) as a "3D bean" and then reused in multiple applications. Each "3D Bean" needs to be hand written using the Java3D library, however once built they can be reused by developers with less experience in 3D development. This research identifies the potential of extending this concept as follows:

> 'As future work we see that the functionality of the Java Bean authoring environment can be enhanced. For instance, it is possible to link it to a database where not only 3D Beans are stored but also geometries without behaviors for the background and textures. The authoring environment could be equipped with all features of a 3D scene editor like texture mapping or lighting specification. Also an authoring environment for building up the 3D Beans themselves from ground up including an animation system from guidance level to task level would be a fruitful complementary effort to the work described in this paper.' [Dorner 00].

Another example of using Java3D in a reusable component style is outlined in [Schonhage 00]. Here a set of "3D gadgets" is described, based on a number of 3D visualization techniques (including Cone Trees). Each gadget is a reusable Java3D element designed for the purpose of visualization in a business context.

### *2.3.3 3D User Interface Libraries*

The example systems and interface features as described in the 3D Interface Applications section (further into this chapter) are all examples of one off stand alone applications. They cannot be used in combination. Ideally the interface features developed in these systems would be available to other interface developers via an interface library or toolkit. The benefits of such libraries have been clearly demonstrated in two-dimensional graphical user interfaces (e.g. Java's Abstract Window Toolkit - AWT).

The widely accepted commercial three-dimensional libraries such as OpenGL and OpenInventor are too low-level (i.e. rendering oriented) to be considered three-dimensional user interface libraries (interface libraries are at a higher level of abstraction , built on top of libraries like OpenGL). However some research oriented three-dimensional interface libraries have been developed in recent years, these libraries can be broadly categorized into 'interface management libraries' and 'interface content libraries'.

#### Libraries for Interface Management

The early three-dimensional interface libraries [Shaw 92][Pausch 93] and [Gobbetti 93] are fundamentally software libraries intended to supply functions to handle the input and output related details of a three-dimensional interface (i.e. early systems were focused on managing the devices and rendering rather the content of the environment).

[Shaw 92] developed a toolkit titled '*MR*', this library introduced the concept *of* '*The Decoupled Simulation Model*', in which the library separates the device and rendering details from the simulation application. [Shaw 92] describes the development of a library consisting of three distinct levels (each level abstracting the user from the detail of the underlying level). The first level works at the device interaction level, the next supplying higher level functions to allow easy manipulation/input from the device level, then the final level which deals with the three-dimensional display and interaction system as a whole (i.e. links up devices/geometry etc and synchronizes them). Synchronization is managed by using a 'generic device' concept, whereby each element in the interface (hardware devices, rendering system, geometry system etc.) is treated as an independent 'generic device', and then a synchronization system merely manages the set of generic devices. Systems such as '*MR*' separate simulation from rendering and interaction, however these systems fail to hide the detail of this separation from the simulation developer (i.e. the developer needs to initialize and manage both sides of the system). In 1993 both [Pausch 93] and [Gobbetti 93] extended the '*Decoupled Simulation Model*', in this way.

[Pausch 93]'s system *'Alice and DIVER*' is based on the underlying separation/de-couple principle (DIVER is rendering/device detail and Alice is the simulation). However the simulation developer need not be aware of DIVER as the Alice system completely hides its detail from the user.

> ' We believe that application programmers should be able to take advantage of this separation without having to explicitly manage the inter-process communication (i.e. Alice to DIVER)' [Pausch 93].

[Gobbetti 93]'s '*Virtuality Builder II*' (VB2) uses an object oriented architecture to encapsulate an application which is composed of a group independent processes communicating via inter-process communication. Each of these processes independently controls a specific element of the interface (i.e. one process for each device, sound, rendering etc). The critical element of this multi-process system is the 'application' process. This process must handle the asynchronous events being received from the other active processes (e.g. device inputs/outputs, rendering, etc), and from those generate outputs as required.

Three-dimensional interface management software continues to be based on this separation principle, programming techniques such as object orientation have enabled the separation principle to be neatly incorporated into an object oriented design where independent 'objects' handle their required part of the complete system.

Following from this research [Gobbetti 93] also identified the need to redirect the focus of research away from hardware and device issues.

> ' The research in virtual environments concentrates far more on the development of new input and display devices than on higher level techniques for 3D interaction' [Gobbetti 93].

## Libraries for Interface Content Development

Early library developers such as [Shaw 92] and [Gobbetti 93], whose focus was on device management features, quickly identified the need for libraries to develop the content side (i.e. what is in the 'virtual world') of 3D interfaces.

> ' There definitely needs to be more research on software development tools for VR user interfaces' [Shaw 92].

[Conner 92]'s '*3D widgets*' and [Gobbetti 93] '*virtual tools*' represent some of the earliest attempts to create interface features that allow user control whilst displaying information about application features.

Research into '*3D widgets*' led to the development of [Zeleznik 93]'s toolkit for developing these widgets. This toolkit is one of the first examples of a toolkit specifically designed for the construction of a three-dimensional interface.

> ' Using this experience, we have developed an interactive toolkit to facilitate the visual programming of the geometry and behavior of such interactive models. The toolkit provides both a core set of 3D widget primitives for constructing interactive behaviors based on constrained affine transformations, and an interactive 3D interface for combining these primitives into more complex widgets' [Zeleznik 93].

This 3D-widget construction toolkit enabled its users to visually[11] combine primitive elements (i.e. points, rays and planes) into three-dimensional structures. In essence this toolkit provides a 'methodology for interactively constructing the geometric behaviour of a variety of 3D widgets and parameterized application objects'. This 'geometric behaviour' defines what information can be displayed and set by this widget. For example a point can display and set a position and a ray can display and set a direction, hence a widget combination of a point and a ray provides a direction marker/setting widget. Although the re-usable components within this system are very primitive (point, line etc), it does support the ability to combine (via linking using a set of constraint relations) existing widgets into larger and more complex structures.

[Stevens 94] proposed a more flexible system for developing three-dimensional interfaces; creating a generic object oriented 'component' concept, which allowed the developer to define both the behaviour and geometric structure of the component. Within this system each object has a geometric structure and also a series of behaviour slots. These slots contain behavioural features (these features are variable values for object settings such as colour, transparency and other features); that in combination with 'interaction techniques' determine how the object reacts to inputs from the user (and what modifications to slots these interactions cause). These 'objects' can be combined into larger structures by relating the slots (i.e. linking or filling them). A good example would be the creation of a three-dimensional menu. Firstly you would create a cube with a text side. Then by creating a separate object/structure to contain a set of these (i.e. one in each slot) you can easily build a menu type component (consisting of the container structure

---

[11] The Visual (non-programming) environment is seen as one of the primary benefits of their research as it enables non-programming users to construct widgets.

with a collection of text-cube objects attached via its slots) which can identify which 'cube was selected'.

In 1994 [Waterworth 94] demonstrated '*VR management Tools*', this system involved constructing a small set of re-usable complex interface items (i.e. 3Dmenus, M-Cubes and SuperCubes).[12] Previous toolkits were designed as flexible systems allowing (and in fact requiring) the developer to build (from primitive elements and actions) any 3D interaction tool. In contrast [Waterworth 94] constructed a small set of complex tools specifically designed for particular tasks, these tools can not be altered to handle other tasks. The [Waterworth 94] concept involves the eventual development of a core set of interaction tools. To be effective this concept requires a set of underlying interface principles (such as the 'desktop' and related principles for 2D GUIs). Unfortunately three-dimensional interfaces are particularly lacking in the area of interaction principles.

Despite this [Waterworth 94]'s research took three-dimensional user interface toolkits from being generic 'build your interface from primitives' systems into a library of re-usable complex interface components (with geometry, actions and controls). Although these tools are not as flexible as earlier systems (e.g. the 3Dmenu can only be personalized in a limited way), they enable a significantly more powerful form of interaction with the user.

The most recent 3D user interface libraries have focused on the development of 'interactive agents' in virtual environments. [Perlin 96] developed a toolkit that functions based on a similar concept to that described by [Stevens 94]; it works by dividing its system into two parts. First is the rendering side that handles the geometric transformations and rendering of the virtual actors. Second is the 'behaviour' side, this side uses a scripting language to allow users to give the 3D character a set of behaviours as defined by scripts (these scripts in turn cause geometric and other actions when someone is using the VR environment). This concept is further extended in [Geiger 00] which identifies some of the key issues involved in taking low level tools and libraries and providing their functionality through higher level tools.

> 'The lack of an appropriate **conceptual model** hinders the design process and renders the communication between stakeholders in the design process difficult. Conceptual design tools adopted from conventional media design (e.g. storyboards) fail to address the special requirements of viewer interaction and dynamic content.
>
> The lack of a **structured design process** makes it difficult for designers and developers to bridge the large conceptual gap between communicative intent and technical implementation. The lack of structure also complicates the development of an appropriate tool-set.
>
> Existing implementations of **visual presentation techniques** are often application specific and non -portable. hindering their reuse, combination and modification.' [Geiger 00].

---

[12] More detail on 3Dmenus, M-Cubes and SuperCubes can be found in the section on 3D Interface Applications.

They propose using an “actor” centred system to solve these conceptual issues. In this system each element in a 3D space is an “actor” and has a role to play, whether that role is interactive, animated or simply structural. This system is implemented along similar lines to those described for [Dorner 00] (involving reusable elements developed by hand and then easily available for reuse). This research offers a different perspective on presenting 3D interface toolkits, with the actor concept providing a mechanism that encourages a more animated form of interface.

Despite the above systems, 3D user interface libraries are currently rarely used for the development of three-dimension interface applications.

Unfortunately the heavy reliance that three-dimensional interfaces have on critical underlying technologies such as rendering hardware, 3D rendering libraries and 3D interface principles, has been a major limiting factor in the development of three-dimensional interface libraries. Existing libraries (such as those mentioned above) are still currently in the very early stages of development, partially due to a lack of broadly adopted interface principles and partially due to the fact that three-dimensional interfaces are still such a new concept.

### *2.3.4 Games and Game Engines*

With the development of new and highly capable 3D graphics hardware in the early nineties [Fuchs 89], the field of 3D interactive computer games began its rise to prominence. Early 3D games such as Wolfenstein 3D and Doom [Wolfenstein 92], [Doom 93] established a new 3D interactive style (i.e. the first person interface) which is still the basis of most games today. These games represent a key step forward in the wider acceptance of 3D user interfaces, however the range of areas in which they can be applied is somewhat limited.

[Chao 01] identifies the fact that 3D graphics has made large steps in the area of game development, however little of that has been applied in user interface design:

> ‘The CHI 97 Workshop on Game Design and HCI suggests that computer game design has much to offer to the HCI community, but little has been done so far’ [Chao 01].

The field of 3D games has blossomed in recent times with many new innovations, including the development and widespread use of game engines (i.e. higher level development tools for game construction) to simplify the development issues involved in constructing complex games. These game engines have enabled a broader range of (less skilled) developers to construct 3D games, by providing a higher level “game engine” application which removes the user from the low level 3D graphics implementation and provides a higher level game design interface [Doom 93]. The advantages that have been made in this field clearly identify the potential for where the field of 3D user interfaces could go, but as yet most of these advantages are untapped.

### *2.3.5 3D Authoring Tools*

This section looks at authoring tools, defined to be higher level tools to enable developers to more easily construct 3D user interfaces. Such tools rely on the existence of underlying

libraries to provide their functionality and usually are interactive representations of the functionality available in those libraries. As demonstrated in the earlier section on 3D interface libraries, it is clear that high level libraries are still in the very early stages of development. This being the case the range of higher level authoring tools is very limited.

Despite this, a range of commercial 3D authoring tools have been developed in recent times, some of which provide 3D interface development features. [Cosmo 98] describes the Cosmo Worlds VRML authoring software application. This application enabled developers to interactively construct essentially static VRML worlds using an interactive 3D environment. Cosmo Worlds had the capability to add interactions in the form of sensors, events, routes and scripts; however the user/developer was required to handle the detail of arranging the events and also was required to write the scripts. This restricted the creation of interactive worlds to those capable of programming. As a result the benefits of the high level interactive world layout tools are somewhat lost due to the lack of high level interaction control tools (i.e. users can shape the environment but many are not skilled enough to add the interactivity and as such still cannot build an interactive 3D environment). However, despite these limitations, Cosmo Worlds brought the possibility of VRML (albeit static) authoring to a broader audience. Despite its apparent potential Cosmo Worlds (and the associated Cosmo Player VRML viewer plug-in) failed commercially (and have been stopped as development projects).

[Holm 02] describes a scenario authoring tool aimed at enabling developers to create virtual reality scenarios (i.e. environments where series of events or scenarios unfold) using a higher level tool. Although this scenario authoring tool is more focussed on “virtual reality” (i.e. device management and control issues) it introduces the concept of enabling “virtual reality” system development without the need for programming (essentially the idea is that the scenario can be altered during the task by the developer to present the user with differing series of events). Unfortunately the method used to avoid programming is essentially almost as complex as the programming itself, requiring developers/controllers to specify and maintain complex relationships between elements as well as specify and control “interface slots” to handle the transfer of information between elements. All of this was to simply organize a path of events let alone arrange the action to take. This interface appears to be useful in its target scenario creation task, as that task involves creating a path or series of events which, once the scenario is started, flow in order to generate the overall scenario situation. However this “in order” style of event system is unsuited to mainstream 3D interfaces where the user can cause any event at any time. Although this system is unsuited to general purpose 3D interface development it does highlight the fact that high level development tools can be implemented to simplify complex tasks of this type.

More recently the Parallel Graphics company has developed a set of Web3D authoring tools [Parallel 03]. These tools are targeted at online product demonstration rather than 3D interface development. However, in their range of products, the “Internet Space Builder” (ISB) application (much like its Cosmo player predecessor) is the most powerful in terms of ability to create 3D interfaces. It offers developers the ability to interactively construct the geometric appearance of the 3D interface. Aside from that ISB offers limited interactive capabilities. Developers are able to attach web links to objects in the world but the task of adding more complex interactions is not broadly supported.

Although these “authoring tools’ offer some interface construction facilities, mainly in the area of laying out the items in the 3D space, they fail to provide enough features (particularly in the area of integrating interactivity and actions) to enable the development of complex 3D interfaces. They are too complex to be used by inexperienced developers and remain the domain of skilled programmers. Currently (with the exception of the builder tool described by this research project) the only way to create a fully interactive 3D interface is to write it by hand either using a low level graphics library or by using a higher level descriptive format with the addition of scripting or programming to handle the interactions.

## 2.4 3D Interface Applications

The earliest three-dimensional interface applications, such as those developed as part of the NASA Ames ‘Virtual Environment’ research in the mid to late 1980’s, were largely rigid three-dimensional environments in which the user could navigate. These systems were extended to support simple interaction in the form of grabbing tasks using a ‘virtual hand’; this simple interaction enabled users to manipulate the location and orientation of the virtual objects.

However, the concept of making the virtual environment into an effective functional interface and more than just a simple immersive experience (of a rigid world) was not largely explored until the early 1990’s. At Xerox Palo Alto Research Centre in 1991 George Robertson, Stuart Card and Jock Mackinlay developed a set of interface applications designed to demonstrate the possibilities of three-dimensional user interfaces. The following sections outline the significant three-dimensional user interface developments from that time forward.

### *2.4.1 Cone Trees*

[Robertson 91] describes the ‘*Cone Trees*’ interface as an ‘*Information Visualization*’ technique specifically designed for visualizing hierarchical structures. ‘Cone Trees’ are hierarchies laid out uniformly in three dimensions.

The top of the hierarchy is represented by the apex of the cone with its children arranged evenly inside the expanding cone shape directly beneath (ie. as the cone widens this allows for more items to be represented, this is ideal for hierarchical information). Further hierarchical levels of information can be represented as additional layers in the cone structure. The system can automatically determine the height between layers by simply dividing screen height by the number of layers. Similarly the cone diameters can be adjusted to ensure that the cone will fit within the room (ie. screen space). The use of partially transparent cones enables the user to percieve the cone structure without blocking his view of the other cones and nodes (file markers).

In addition to this structural representation of the information, the ‘cone trees’ system also uses animation to aid in representing the information. When a node is selected the cone tree rotates so that the node and all nodes in the path to it are brought to the front and highlighted (they proposed 2 highlight methods: colour change and text activation).

Text representation can be problematic due to the vertical orientation of 'cone trees'; this problem led to the development of 'Cam Trees'. 'Cam Trees' are essentially cone trees rotated by 90º, such that the apex is on the left side and the cone expands out towards the right side of the screen/room. This horizontal orientation can more easily display the textual information for the nodes.

Aside from the basic concepts in the 'cone trees' interface it also offers several additional operations which can be used on the cone trees.

- ***Pruning and Growing*:** refered to as 'gardening' operations, these involve giving the user the ability to interactively close (or prune) and open (grow) sections of the hierarchy. For pruning the user simply flicks a node towards the root and its descendants are hidden. For growth the user simply flicks the node away from the root and its descendants are displayed.
- ***Hierarchy Alteration*:** the user is able to grab a node and drag it to a new location in the tree system.

The cone trees interface is particularly effective for visualizing hierarchical information because:

- Most Hierarchies in real applications tend to be broad and shallow, such information is impossible to fit onto a two dimensional screen. However in three-dimensions (such as cone trees) the interface can use depth to fill the screen with more information.
- The selected path is more easily identified due to its animated movement to the front position and the path highlighting.
- Animated tree-rotations provide a visual method of altering the users perception of the underlying hierarchical structure.

[Calitz 01] demonstrates the use of the Cone tree concept for application in displaying hierarchical data on the web. In addition, this research discusses the concept of creating an object oriented, data driven framework for the creation of such Cone Trees (i.e. taking raw hierarchical information and using the frame work to fit it into a cone tree representation) and thus creating a re-usable framework that could be applied in other applications. [Xiaoguang 99] and [Tanriverdi 01] both discuss similar object oriented frameworks/methodologies not specifically for cone trees but for the general implementation of virtual reality systems.

### *2.4.2 Perspective Wall*

[Mackinlay 91]'s '*Perspective Wall*' is another information visualiation technique developed at Xerox in 1991. The perspective wall is designed to present linear information (such as text) by smoothly integrating detail with contextual information in a three-dimensional interface.

The interface of the 'perspective wall' involves the use of a three-sided wall; this wall has a central panel (directly in front of the user) onto which the detailed information is presented. The two side panels are used for the display of contextual information. The user is able to adjust the relationship between detail and context via a set of animated

actions that the wall supports. The most effective action is the '*View Transition'*, where the user is able to adjust the contents of the detail wall by selecting a new item (from within either the detail or context walls). This action causes the new item to 'scroll' into its new position in the centre of the detail wall.

### *2.4.3 Information Visualizer*

[Card 91]'s '*Information Visualizer*' contains a number of interface elements, each of which handles a particular three-dimensional interface issue (ie. the Information Visualizer incorporates: 3D Rooms, Cone Trees, Perspective Walls, Data Sculpture and the Spatial Office Building). The top level concept is that of '3D Rooms'. This concept is loosely based on the 'rooms' system of two-dimensional graphical user interfaces. This two-dimensional concept involves the creation of separate interface screens (ie. multiple desktops) containing differing information, in essence it is an attempt to give the user a larger working space by providing a set of separate screens onto which the user can arrange his information.

The '3D Rooms' system uses a similar concept, the difference being that the separate areas are three-dimensional rooms (or spaces) rather than 2D screens. Within the global three-dimensional space the user can move around between rooms to access the different information inside the rooms.

The information inside the rooms (and its presentation) is the other focus of the '*Information Visualizer*'. In addition to '*Cone Trees*' and the '*Perspective Wall*', the information visualizer describes two other methods of representing and interacting with information.

'*Data Sculpture*' is a method of scientific visualization enabling the user to view complex data presented in a visual form.

The '*Spatial Building*' provides an interface based on an office building, where the user can select 'offices' from within the 3D building, in doing this the user can obtain information on the location and occupant of the office.

The 'Spatial Building' effectively uses a three-dimensional interface to convey three-dimensional location information to the user. It is not unlike an interactive three-dimensional map.

Although these are effective methods for visualizing specific types of complex data, the 'Data Sculpture' and 'Spatial Building' interfaces do not offer the generic information representation advantages demonstrated in 'Cone Trees and Perspective Wall'.

### *2.4.4 Racks (Transformation Tools)*

The goal of [Snibbe 92] was to develop 3D widgets for the performance of a simple three-dimensional interactive task, in this case widgets to apply high level deformations to 3D geometric objects. These three-dimensional deformation tools are called 'Racks'.

A "Rack" is a 'line' (thin cylinder) with 'handles' (also thin cylinders at 90 degrees to the main cylinder but with knobs at their top) hanging off it. These handles can be turned to

deform the shape on the rack, like running a pin through something and using the handles to deform it.

Various object manipulations can be achieved with these tools (i.e. the colour of the handle indicates its function (red for twist, blue for scale etc.). In the development of these ‘Racks’ [Snibbe 92] identified the following issues in widget design:

- **Self Disclosure**: A widget whose geometry indicates its behavior is called 'self-disclosing’ (e.g. a handle that twists an object can itself be a twisted object, thereby suggesting its action)
- **Implicit vs. Explicit control of Parameters**: a widget should be allowed to interactively make implicit adjustments without interrupting the user (e.g. increase tessellation as the user increases an objects complexity)
- **Degrees of Freedom**: A tool can be made more effective by removing unnecessary degrees of freedom with constraints
- **Intended Use**: Ensure that the tool meets the usage requirements (i.e. an artistic tool does not need numerical details where an engineering tool might)

### *2.4.5 3D-Menu, M-Cube and SuperCube*

In an attempt to develop a three-dimensional equivalent to the widely used two dimensional menu, [Waterworth 94] developed three tools capable of handling menu type interactions in three-dimensional space.

#### 3D-Menu

The 3D-Menu is basically a direct copy of two-dimensional menus with an added depth dimension (ie. each menu item is a box in 3D space with textual title on its facing side). These menus cascade by having the current menu push earlier ones backward into the three-dimensional space. They also use a colour scheme to indicate buttons (menu-items) belonging to the same hierarchical level (ie. each hierarchical level has its own colour thus making it easier to identify the level a menu-item is at).

#### M-Cube

The M-Cube is a cubic representation of a set of menu-items. That is, each of the six sides of a cube represents one of the available menu-items. This tool is more effective in three-dimensional space as it does not require the cube to be oriented in any particular way when presented to the user (unlike menus which must be upright and front on to the user). At any given time the user can only view three of the cubes options. However the user is able to rotate the cube by clicking on one of its corners, thus the cube rotates in that direction to show the hidden options. When a selectable item is the equivalent of a cascading menu, this is indicated by a bulge on the corresponding M-Cube face. Selection of that face results in the replacement of the existing M-Cube with the new. To move back up the hierarchy the user selects the corner of the cube nearest to his view. Colour coding to indicate depth within the M-Cube hierarchy is again used in this system, as is the automatic rotation of text labels in order to make them appear upright.

### SuperCube

The SuperCube is a direct extension of the M-Cube. Where the M-Cube is limited to six options at any single hierarchical level, the SuperCube avoids this problem by not attaching menu-items to a cubic face. Instead it uses the available faces of the cube to scroll though an arbitrary length list of options. This scrolling is achieved by taking the rotation action as used in the M-Cube and replacing it with a simple shift in the list of options. For example rotating the SuperCube by pushing away the left edge of the nearest face corresponds to moving back through the list of options. Pushing away the right edge of the SuperCube causes the SuperCube to move forward through the options, pushing up the upper corner of the upper face corresponds to moving up the hierarchy of options.

This system effectively demonstrates the use of a six-sided cube to replace menu interactions. The SuperCube overcomes the six-side limitation, however other possibilities including the use of more complex polyhedra (eg. dodecahedron: 12) may offer a more effective selection mechanism.

## *2.4.6 HoloSketch*

HoloSketch [Deering 95] was developed with the goal of proving that a high quality, high accuracy VR system based 3D object editor should be both easier to use and more productive than traditional 2D projection systems. Despite its goals being directed towards proving a hardware related issue; the HoloSketch project developed several effective three-dimensional user interface application elements. In particular it's innovative three-dimensional menu-like system.

In two-dimensional interfaces an array of screen buttons or menu selectors are usually attached to one side of the drawing space (e.g. the menu bar at the top of the screen). For a three-dimensional interface this is a problem as the user keeps changing his view and a permanent 'menu' would block out a large area of the view. HoloSketch provides all of the main controls on a single, huge pop-up menu which is engaged by clicking (and holding down) the device button. [Deering 95] describes this as a 'fade-up' menu, due to the fact that when the menu is activated the existing work fades into the background colour, leaving only the menu system to view. This is done for two reasons. Firstly it improves rendering performance as the existing work does not need to continue being rendered and secondly it allows the user to focus on the menu system.

The menu is organized as a 3D pie-menu located around the pointer location in 3D space. The menu remains active until the user releases the device button (it then fades back out). Selecting items within the pie-menu is as simple as pointing at them (visual feedback in the form of 'lighting up' indicates which item the pointer is on).

This system handles sub-menus in a similar way to that proposed by [Waterworth 94] (i.e. push older menus back to create space for sub-menus). When a menu with sub-menu elements is activated the whole pie-menu moves back into the screen and the sub menu (also a pie-menu) fades into the front position. The pie-menu system described here was extensively used in the development of several animations, and the users found it an effective selection mechanism.

### *2.4.7 Visualization Techniques*

Three dimensional systems have been used to visualize large scale complex structures for some time. The early 3D interface applications (eg. Cone Trees) were primarily aimed at this task. Although these visualization systems are often not as interactive as full 3D interfaces they do provide an interesting insight into how data can be most effectively presented in 3D space.

[Graham 95] describes an early 3D visualization technique used to aid the visualization of databases and their contents. The challenges in such 3D visualization tasks are many and involve not only understanding the information to be presented but also the potential and limitations of the 3D presentation mechanism itself.

> 'Without good design and real consideration about the interaction issues, database visualization will be little more than a 3D table, and not the "matrix of awareness" that we wish it to be.' [Graham 95].

[Wakita 03] proposes several new spatial visualization techniques, each aimed at visually presenting large data sets (in particular representations of the internet) using three dimensional methods to enable viewers to gain a better understanding of the information space. Essentially these visualization techniques attempt to present the information in terms of user activity relative to the information (i.e. how users are interacting with web/internet information sources). [Wakita 03] defines two major directions for structures used to visualize information, those being:

- **Replicated Structure**: Visualizes items and phenomena as they are in the actual world. This taps into the users existing experience and makes the structure intuitively easy to understand.
- **Creative Structure**: Visualizes items and phenomena in a totally different way from their original state. This measure is generally used to systematize knowledge and concepts into a form that is more easily understood.

In using these concepts they propose three methods for presenting large data sets. The first method, City Tomography, is essentially a replicated structure, based on a real world city with buildings and streets. Using this city layout users are able to navigate the city scape accessing information (such as details about various shops etc) which is displayed on the walls of the buildings. The next two systems are creative in style with the INFOTUBE using a tube structure to present information about a collection of stores. Essentially each stores information is displayed as a cell in the tube structure.

The third system, Ryukyu ALIVE, is used to present data relating to web pages and their access rates by users (i.e. ALIVE stands for Access Log Information Visualizing Engine). In visualizing this data Ryukyu ALIVE creates a 3D galactic space made up of a set of stars/icons (each representing a separate web page). The visualization takes pages that have been accessed most recently and places them at the outer edges of the spinning galaxy. Pages that have not been accessed recently slowly work their way in to the centre. Hence the system enables viewers to quickly see which pages are being accessed and which are not.

### 2.4.8 Data Mountain

The data mountain [Robertson 98] is a 3D interface designed as a document management system. It is essentially a simple three dimensional inclined plane onto which the user can position items/documents. The 3D nature of this system taps the users spatial memory capabilities (i.e. the ability to remember where you put something). Implementing a simple placement method this interface enables users to move items to any location on the "data mountain". To aid this movement task the mountain provides "active page avoidance" where other pages move out of the way of the page being manipulated. Finally it incorporates audio and visual region queues to aid in managing the items on the mountain. This method proved effective for the task of item placement and retrieval with the general observation that the freedom of placing items anywhere on the plane made it simpler for users to relocate those items when needed later. [Cockburn 01] followed the initial data mountain research with a separate analysis in which user trials indicated that there was no significant advantage in terms of performance when using 3D presentations as compared to 2D systems. However in general users found the 3D systems to be more enjoyable/preferable.

> 'They show no significant difference between task performance in 2D and 3D, but a significant preference for the 3D interfaces.' [Cockburn 01].

### 2.4.9 The 3D Workspaces

[Leach 97] introduced the concept of **The Tunnel** interface. This interface uses a tunnel like structure to display 2D windows either on the walls of, or displayed centrally in, the tunnel. In addition to this window layout feature the interface also uses a console displayed at the bottom of the "floor" plane near the users location. This console enables settings to be made to the overall tunnel (i.e. layout etc). This workspace provides an interesting new method for manipulating a window based interactive environment.

**The Task Gallery** [Robertson 00] is a window manager that extends the concepts of [Leach 97] and uses interactive 3D graphics to provide direct support for task management and document comparison, provided through the use of the 3D workspace. This workspace displayed in the form of a 3D room enables tasks to be displayed within that 3D volume.

User tasks appear as artwork hung on the walls of a virtual art gallery, with the selected task on a stage (much like the GUI desktop). User studies suggest that this Task Gallery helps users with task management (i.e. the use of the walls to identify differing tasks makes it easier to manage), is enjoyable to use, and that the 3D metaphor evokes spatial memory and cognition. Further studies have indicated varying levels of usefulness with [Cockburn 02] indicating that:

> 'Our results indicate that for relatively sparse information retrieval tasks (up to 99 data items), 3D hinders retrieval.'

In addition to the basic workspace this research also uses the concept of toolspaces as described in [Pierce 99]:

> ‘The existing Windows desktop metaphor uses menus (especially the Start Menu) and toolbars to give the user access to commonly used tools and documents. To better fit the metaphor of moving through a hallway, we designed the Task Gallery so that the user carries tools and documents associated with the virtual body, using an adaptation of Glances and Toolspaces. Glances are a lightweight, ephemeral way of looking around in a virtual environment without moving the virtual body. Toolspaces are placed around the user, and hold various tools or objects, keeping them associated with the virtual body as it moves through the virtual environment. The Task Gallery has toolspaces left of, right of, above, and below the user. Hand and feet are shown to make the scale of the objects in the toolspaces more obvious and to suggest that these tools stay with the user as the user navigates through the environment.’ [Robertson 00].

The toolspace is an interesting concept enabling the user to carry items with him/her through the world. In this system these items are loosely arranged (no particular organization or structure, merely a space to fill), however as the researchers identify there is potential to implement more effective storage mechanisms:

> ‘ We believe that there are several potential avenues for future work. One is to explore other layouts and constraints for placing objects in toolspaces, especially if large numbers of objects need to be stored in a small number of toolspaces. We believe that different tasks will require different layouts and constraints, and determining the correct design choices will require careful experimentation’ [Pierce 99].

Other workspace type concepts of this kind include [Cubaud 01,02], which describes the idea of a similar primary workspace with peripheral 3D information (note that this is not actually implemented as a functional system). In this case the task involves searching for and manipulating items from within a library collection.

One of the problematic issues identified in virtual 3D environments is the fact that often one item will obscure the users view of another. [Bell 01] proposes a view management system to ensure that items in the world do not cause this problem, this research represents a new element in virtual reality systems, having moved from the early systems which focussed on separation of the physical input/output and content issues, to a broader focus on the content and how that can be effectively arranged and even manipulated after creation (as described in [Bell 01]) to avoid some of the known pitfalls in 3D environments.

There are several notable interface applications which have already been covered in preceding sections, these include ‘*Worlds In Miniature*’ [Stoakley 95][Pausch 95] in the Navigation Principles section, ‘*Silk Cursor*’ [Zhai 94] in the See-through Interfaces section and the ‘*Aura*’ [Fahlen 93] based application in the Multi-user Principles section.

## 2.5 Summary

As this background section demonstrates, three dimensional graphics is now becoming far more mainstream. Areas such as game development already make extensive use of the potential of interactive 3D graphics. However the use of 3D graphics in functional user

interfaces remains largely untapped. The necessary hardware performance is there but the software systems and interface principles are largely underdeveloped. With the exception of a few examples we have seen only a very small number of effective 3D interface applications.

> ‘There is a scarcity of effective, practical Web3D applications. There appears to be several reasons for this phenomenon. First, there is a general lack of understanding among developers of how to use and integrate 3D graphics effectively for various problem domains. In other words, it is unclear as to what types of tasks and users are best served by presentations that incorporate 3D graphics with other representational forms. Secondly, integrating 3D graphics into presentations is potentially time and resource intensive, due to the limited availability of integrated authoring tools.’ [Zimmerman 03].

Three dimensional interface development tools are getting better, but they remain the field of highly skilled graphics programmers. There is a clear lack of easy to use interface construction tools, the kind of tools that can be used by mainstream developers. In simple terms several innovative/interesting systems have been described, however 3D interfaces have yet to establish a permanent foothold in mainstream user interface applications.

# *Chapter 3: Challenges for 3D Interfaces*

The application of three dimensional graphics to the area of user interfaces offers enormous potential and with that potential comes a set of equally enormous challenges. The biggest challenge for the field of 3D user interfaces is in taking the potential offered by a 3D workspace and converting it into a functional and effective system. Some areas of 3D graphics have seen a great deal of success. For example 3D in the area of immersive experience systems has been effective (i.e. VR systems that enable the user to walk around in a computer generated 3D environment or flight simulation systems). However significantly less success has been achieved in making 3D systems that work effectively in other more functional tasks.

As the previous section outlines, much of the early research in 3D interfaces deals with the hardware and device related issues (i.e. virtual reality systems) and this area remains a strong research field. This PhD research project is interested in the functional content of the world and how to make the 3D workspace effective for achieving real world tasks rather than the hardware related issues.

Knowing that 3D graphics has been widely adopted in areas such as game development and animation, shows its potential. However effectively applying 3D graphics to user interfaces remains the key challenge. In dealing with this challenge there are a number of areas that need to be addressed including:

- Determining a basic set of principles for 3D Interface Design
- Making 3D Interface Development available to a broader population
- Developing 3D Elements/Systems to handle common real world interaction tasks

The sections that follow cover each of these challenges and the issues involved in greater detail.

## 3.1 Tools for 3D Interface Development

One of the biggest current challenges for 3D interfaces is the need to enable more people to develop such systems. The fact that developers currently need to have both strong programming skills as well as an excellent understanding of 3D and user interface concepts limits the number of developers capable of implementing 3D user interfaces.

Overcoming this issue requires the construction of tools that will enable less experienced users to enter the field and begin developing 3D interactive systems. Achieving this will require the development of powerful, yet easy to use, construction tools designed to hide the complexities of the 3D graphics and interaction systems and allow the user to focus on the higher level interface design issues.

### *3.1.1 How are 3D Interfaces built today?*

A three dimensional user interface is a complex system and requires a significant amount of development effort. Faced with the challenge of constructing a 3D user interface, what steps would current developers take to implement such systems?

The answer to this question depends a great deal on the level of skills that our developer has. For example the following list shows the differing approaches that would be taken by developers of differing skill levels:

- **The Experienced 3D Developer** (able to program in various languages and has experience with low level 3D graphics libraries such as OpenGL).

  Would write a program using a suitable programming language (eg. C++) with a low level 3D graphics library (eg. OpenGL) to implement the required system.

  or

  Would use a world description format (eg. X3D) and the necessary scripting/programming to implement the interactivity. This would take a *significant amount of development effort and time* but would generate an *effective interactive 3D interface.*

- **The Mainstream Developer** (able to program in various languages).

  Would find this task difficult and would need to learn a set of new skills to effectively implement it (i.e. needs to learn a graphics library such as OpenGL or learn to use a format such as X3D/VRML with scripting). Without learning these new skills the mainstream developer *could not implement an interactive 3D interface*.

- **The Experienced Tool User** (experienced user of software tools, but cannot program).

  Would find this task difficult and would only be able to use simple static world construction tools to build an essentially non interactive system. Without learning a large set of new skills the tool user *could not implement an interactive 3D interface*.

- **The Inexperienced Developer** (has little or no experience with either tools or programming).

  Essentially this task is far beyond an inexperienced developer. Consequently an inexperienced developer *could not implement an interactive 3D interface.*

As the above list demonstrates, the ability of a developer to create an interactive 3D interface is very dependant on the availability of suitable tools. For example the experienced 3D developer is able to make use of the low level 3D graphics libraries such as OpenGL to build the interface. For this user his/her specialist skills enable him/her to take these low level tools and use them to build a more complex system. Unfortunately the current range of development tools does not effectively provide for a very broad range of users, in fact it really only supports the top end (i.e. experienced 3D developers). Given that the experienced 3D user group is actually only a tiny proportion of the overall population, it is apparent that the development of 3D interfaces (by the broader population) is very much restricted by a lack of suitable tools.

In the current environment the tools that are available for 3D interface development tend to be low level (i.e. difficult to use and requiring a lot of specialized skills) programming type tools, including:

- Low Level 3D Graphics Libraries (eg. OpenGL, Direct3D etc)
- Higher Level 3D Graphics Libraries (eg. OpenInventor)
- High Level Graphics Formats (eg. VRML / X3D)

The fact that these tools all fall into a category requiring high levels of programming skill clearly identifies the key problem (i.e. the lack of higher level 3D interface development tools). This lack of high level tools in turn leads to a small developer base and as such limits the growth of 3D interfaces as a field. In a way this is something like a chicken and egg situation, because users need powerful yet easy to use tools before they begin developing 3D interfaces. However at the same time tool developers need to know what features are needed, and this information is not known until more example systems are built.

### *3.1.2 Steps to further 3D Interface Development*

The task of improving the development of 3D interfaces is subtly different from the task of developing the interfaces themselves. Improving development facilities involves looking at enabling people to be more than simple users of 3D interfaces, but instead be capable of constructing these interfaces themselves. The objective is to empower these users and provide them with the tools they need to begin creating new and innovative 3D user interfaces.

As the previous section outlines it is clear that the current set of available 3D interface development tools do not provide an effective system to enable most developers (or potential developers) to begin working in the area of 3D interfaces [Zimmerman 03]. With this thought in mind it is clear that the next key step required is to develop a better set of 3D interface development tools.

The tools that are available today provide all of the necessary functionality (eg. OpenGL incorporates all of the display and interaction capabilities that a 3D interface requires), however they are not appropriate for the task. This is not due to a lack of functional capability but instead is due to a lack of accessibility (i.e. they are only accessible to a limited set of skilled developers). For example VRML[13] provides all the necessary functionality (i.e. displays 3D worlds, enables interaction sensors and actions), the problem is that to develop an interactive system using VRML requires the developer to hand write a complex combination of VRML and scripts. Essentially it is simply too difficult for most developers to use this system.

In an ideal environment developers really need an interactive tool that enables them to create their world from a simpler perspective; not needing to deal with issues like geometry, sensors and scripts. Instead the developer would like to focus on the higher level task of user interaction with his/her system. Such an ideal system would provide the user with the ability to simply drag and drop 3D components into place to handle specific interaction tasks, in the same way as the 2D GUI interface building tools allow today.

---

[13] OpenGL programs could be used as an example here. But there is even more complexity in such systems, so the simpler VRML based example is described.

Although a system of this type sounds very appealing, it is still some way from being achieved. There are many challenges that must be overcome to make such a system work, some of those critical challenges include:

- Identifying key interactive tasks and developing specialized 3D components to handle them
- Identifying key design principles for application in 3D interfaces
- Enabling simple construction of 3D environments
    - Allowing the user to add/edit objects, lights and cameras without needing to delve into the details of how they are implemented (eg. let the user simply drag and drop items into the space).
    - Hiding detail of 3D display system
    - Hiding detail of 3D Interaction mechanisms (i.e. no need for the user to be involved in low level mechanisms like picking or sensor controls).
- Providing the user with a simple means of controlling interactive behaviour (i.e. take away the need to write programs or scripts in order to have interaction).
- Providing the user with a simple means of specifying actions (i.e. as above - take the need for programming actions out of the development process).
- Bringing these elements together into a cohesive "construction tool"

The development of a tool of this kind would certainly represent a large step forward for the development of 3D interfaces. Creating such a tool (or even a subset of this larger overall goal) would provide developers with a more effective set of development options and enable more users to begin developing 3D interfaces. It is in this area that this research is most directly focussed (see the later sections on "*The 3D Interface Construction Tool*" and the "*Set of 3D Interface Components*" for a more thorough discussion of this project and how it addresses these issues).

## 3.2 Principles for 3D Interface Design

Another key challenge for 3D interfaces is the discovery and use of design principles. These are essentially guidelines for how to make the most effective use of 3D space in an interactive interface. The fact that 3D interfaces are such a relatively new field means that principles and guidelines of this kind are in relatively short supply (see section "*3D Interface Principles*" in the *Background* chapter above). However as more and more systems are developed, these guidelines continue to emerge providing developers with a base from which to refine future systems.

### *3.2.1 Design Principles in Use Today*

Sadly there is no master set of design principles that can be applied for 3D interfaces today. This is unfortunate because principles bring with them consistency which makes development easier. Despite this lack of principles the field is starting to see a set of basic principles of this type beginning to develop. Such principles are usually identified through the laborious trial and error process of testing interactive 3D systems on real

users. As 3D interfaces have been in existence for only a short period of time there have only been limited opportunities to identify such design principles. However 3D interface developers are not totally without guidelines. Early user studies and their results provide some interesting facts and principles (eg. [Eisenberg 97] etc). These studies and the concepts they identify provide a very basic set of guidelines, however this set needs significant development to reach a functionally useful stage. For more information on existing 3D interaction principles see the earlier section on *3D Interface Principles (pg 8).*

The early 3D interfaces tend to be unique one off systems in which the basic functionality is the primary issue rather than broader design principles. The field of 3D interface development is yet to see interfaces in which design principles are applied as part of the development process; instead the early research systems are designed more with the objective of identifying principles rather than applying existing sets. Although this lack of widely accepted principles makes it more difficult to design systems, it also leaves open the possibility of identifying new principles in a range of areas. This research project, through its components and user testing, contributes in this area through a range of interesting findings.

## 3.3 Applying 3D Interfaces to Real World Tasks

The reality is that people, as humans, live and function effectively in a three dimensional world. Given that this is the case, it is obvious that 3D interfaces can be applied to almost any interactive task. In fact there is virtually no interactive task that will not have an equivalent in the “real 3D world”. Yet there has been only limited success in the area of functional 3D user interfaces. With the large range of potential, yet untested, concepts and components, 3D interfaces represent one of the areas with the biggest untapped potential in user interface development.

### *3.3.1 Which 3D Components are Effective?*

The background section earlier in the thesis covered a range of existing systems which have proven their effectiveness for particular tasks. Examples include systems such as cone trees for visualizing and interacting with hierarchical structures [Robertson 91] (i.e. representing the kind of structures used in most OS file systems). These systems make use of the extra dimension that a 3D interface has to offer to provide a new way of looking at existing interface issues. However, despite these successes not all “real world” tasks are suited to 3D interactive systems. For example it is clear that some tasks are inherently unsuited to 3D systems. The classic example here is that of two dimensional text. The very nature of the item makes it less than ideal for a 3D interface. That is not to say that you cannot simply insert 2D items such as text into 3D spaces. However what it does mean is that there is little that can be done with text where the extra dimension can be particularly useful.

Aside from these few cases, such as text, where 3D cannot extend their capabilities, most interactions can benefit from the effective use of three dimensions. In particular the extra dimension would most likely be effective when displaying large bodies of information where spatial relationships can better represent the information than would be possible

with two dimensional systems (eg. in fact visualizations of large data sets has already been implemented in a number of forms [Robertson 91], [Wakita 03] etc).

Given that modern networking has enabled us to access such huge bodies of data through the Internet it would seem likely that 3D interfaces could be most effectively applied in this area. Examples might include displaying search results, searching libraries of data or many other interactive real world tasks. The key challenge, that remains to be addressed, is applying the potential of three dimensional interfaces in the effective implementation of functional real world tasks.

# *Chapter 4: Overview of the 3D SPACE Project*

## 4.1 The Goals

The initial goal for this research was to apply the advantages offered by a three dimensional system to the area of user interfaces. In particular focusing on the development of the content of the 3D world (i.e. the interactive pieces that enable a user to achieve functional tasks from within a 3D space). Much of the previous research in 3D user interfaces has focussed on the hardware and device issues; however this project's goal was to focus on the content of the world to make it more capable of providing useful interactions. The aim was to not simply immerse the user in a 3D environment, but to make that environment rich in interactive tools enabling the user to complete a range of real world tasks all from within the 3D space.

Following the study of previous work it was apparent that, in addition to a lack of established "3D components" for common interaction tasks, there also exists a lack of development tools for the creation of interactive 3D user interfaces. This lack of established components and tools is clearly a limiting factor in the growth of 3D interfaces in general. Faced with these challenges the goals for this research project focused more specifically on the area of 3D interface development tools. In particular focusing on developing the area relating to high end tools that enable simple interactive construction of 3D interface applications. In implementing this goal there are two key elements that were the obvious primary focus of the research:

- High Level 3D Interface Development Tools
- Reusable 3D Components for Common Tasks

To achieve these goals, and develop an effective 3D interface development platform, would provide an environment from which more developers could undertake research in 3D interfaces. This would enable larger, more complex systems to be implemented through the availability of powerful interface construction tools incorporating reusable "components" designed for common interaction tasks.

The first objective is to design and create a set of demonstrably useful tools (or interface components), to facilitate the 3D implementation of some mainstream "real world" tasks. This involves the design, construction and testing of a range of interactive 3D "components" on real users to determine how effective or otherwise they are.

The second objective is to build these tools so that they can be easily used/re-used for various applications (i.e. do not build a one off solution for a specific set-up, instead build a set of generic tools/pieces (like a library of interface components) that can be re-used and pieced together into larger items to solve various problems).

As virtual reality and 3D environments in general are still young areas of research, it is apparent that at this point there has been little research into what does and does not work. This is particularly true in the area of interactive, functional tools inside a virtual environment (see previous section on *Background* for more detail). Given the lack of

existing research, the key sub-challenges to be addressed in achieving the overall goal were:

- To establish a set of basic 3D interaction principles
- To create “3D Components” for common interaction tasks
- To create a library of re-usable “3D Components”
- To apply the “3D Components” to real world tasks and test their effectiveness
- To create a development tool enabling rapid development of 3D interfaces

## 4.2 The Approach / Methodology

The basic approach used in this research is very simple and involves the development of new 3D components to handle common interaction tasks. To simplify this development process a 3D interface builder tool/application was developed to make it possible to quickly build and alter these components. Consequently the methodology implements the following steps:

- **Build the Base 3D Interface Builder Tool** - construct an application to enable development of interfaces and components from either basic elements or more complex pre-defined components.
- **Select an Interaction Task** - choose a common task for which to provide an interface component.
- **Design 3D Components to handle the task** - Using the inherent advantages of three dimensions design components (many options can be implemented) to handle the specified task.
- **Build Each Component** - use the builder tool to construct the components (where possible build components so that they can function in a generic reusable form as this enables them to be incorporated into the builder tool itself later).
- **Test Components On Users** - run trials using the new components to determine their effectiveness on real users undertaking real tasks.
- **Analyse Test Results** - compare the performance results of the new components. This is critical in establishing which are effective and which are not. Note that sometimes it is possible to learn a great deal from failed components (i.e. they may establish new and often valuable principles).
- **Incorporate new “Components” into the “Builder Tool”** - Components that have proven effective for their task can be incorporated into the “builder tool” (in generic form) so that they are available for reuse in other 3D interfaces.
- Return to the “**Select Interaction Task**” above.

The general approach described above provides both a mechanism for creating and testing new 3D interface components and also for developing a tool to enable the rapid construction of 3D components and interfaces. The development and testing of the components themselves contributes to expanding the capabilities of the builder tool.

Essentially one tool builds the other and vice-versa. Where possible the components are designed as reusable tools to allow for future usage. The general approach to component design is to solve problems using small components which can be combined into larger systems.

## 4.3 The Broad Conceptual Ideas

This research is based on a couple of fundamental ideas in a range of differing areas. For example the design, construction and use of the 3D interface builder tool is essentially based on the concept that by providing effective tools for interface construction, this will:

- Simplify the task of 3D interface development
- Encourage more developers to build 3D interfaces
- Enable developers to create larger and more complex systems
- Enable more rapid development of 3D interfaces
- Create more reliable and consistent 3D interfaces
- In general take the focus of 3D interface development away from the low level graphics programming detail and bring it up to the higher level interface design

Essentially the idea of the development tool is to attempt to provide a drag-and-drop style 3D interface construction tool to enable developers to make use of existing 3D components to rapidly develop functional 3D user interfaces.

In the area of 3D component design, the key concepts that this research pursues include the concept of active interfaces, that is interfaces that are not simply static waiting for the user to interact but actively present their data or information (eg. items that throb or spin to attract users attention to their function). This and the following concepts make up the key backbone of the conceptual design being tested in this research:

- **The Use of Motion in 3D Space**: The human eye is naturally attracted to motion in the world, yet most of our interfaces make little use of motion in displaying information. The concept of viewing information as a flow or elements as active items appears to have potential.
- **Independent User**: This concept presents the user in a 3D environment as a mobile independent entity (whether the world be a local machine or the internet; the user interacts as a stand alone entity, needing to take everything he/she needs with him/her - perhaps in the form of a tool belt with tools, or Swiss army knife type concept).
- **Active World**: World as an “active” environment where information interactively presents itself to the user, rather than statically waiting for the user to interact with it. Examples of this concept include the idea of set of data/information as a flow of items.

In general all of the components are intended to make the most effective use of the spatial nature of 3D environments. It is this use of three dimensional space that is their primary design issue.

## 4.4 Overview Summary

This research (i.e. the 3D SPACE project) is primarily intended to advance the field of interactive 3D user interface development. Through the development of new and powerful interface construction tools, as well as the discovery of new interactive components and principles, this research aims to bring 3D interface development to a new, broader audience of developers, whilst at the same time making development easier and more efficient for existing developers.

With the construction of the high level “3D Interface Construction” tool this research enables less skilled (i.e. non programmers) to develop highly interactive 3D interfaces. This contributes to the field of 3D user interfaces through its ability to bring a whole new collection of developers to the field and also through its ability to simplify the task of interface development in general.

In addition this research contributes in the area of component design and creation with the development of a set of new 3D interface components, including the flow, 3D slider, drum, circulatory system and others. Each is a generic reusable element which handles a particular common real world task and can be easily accessed for reuse through the construction tools component oriented design.

# *Chapter 5: The 3D Interface Construction Tool*

In developing the area of 3D interactive systems it appears that the availability, or lack thereof, of development tools is a key stumbling block to bringing 3D interfaces to a wider audience. This section describes a construction tool, developed as part of this research, designed to provide a development platform from a which a broad range of 3D interfaces can be easily constructed.

## 5.1 Why Build a Tool?

The development and use of tools is one of the key strengths of human nature. This is particularly evident over recent years in the area of 2D GUI development. With the creation of powerful drag-and-drop style GUI builder tools the construction of 2D interfaces has moved from complex hand developed pieces of graphics programming (in the mid-late 1980's) to becoming a simple interactive drag-and-drop task today. In many ways the complex process required to develop those early 2D GUI interfaces is not dissimilar to the challenges faced in constructing 3D interfaces today.

The use of tools in building interfaces of all kinds provides a range of key benefits including consistency, reliability, efficiency and, not least of all, the ability for tools to hide the complexity of the resulting system, thus making it easier and faster to develop the desired interface functionality.

For the area of 3D interface development, the concept of an easy to use rapid interface development environment is an appealing one. Such a system would enable skilled users to develop complex systems significantly more quickly than by programming them by hand. At the same time, such a tool would enable a new group of developers to enter the field of 3D interface construction. These developers did not have the necessary graphics programming skills to build a 3D interface previously, however with the availability of a high end construction tool (i.e. one that hides the complex programming detail) they can now develop such systems.

In simple terms the answer to the question posed by this section (i.e. "Why build a Tool?") is that a tool makes it possible to achieve outcomes in 3D interface development that would be beyond our reach without it. In the same way that the task of nailing together two pieces of wood is a difficult if not impossible task with your bare hands; the task of building an interactive 3D interface is also a difficult, if not impossible task for many interface developers. Yet in both these examples a suitable tool turns these tasks into simple challenges which can be completed by almost anyone. It is this ability of tools to make complex tasks simple that is the primary reason for building a 3D interface construction tool.

## 5.2 The Features of a 3D Interface Builder Tool

The concept of a “development tool” is a relatively broad one. For example this could mean a low level programming library, such as OpenGL, or it could mean a higher level interactive drag-and-drop style tool. Identifying the desired level of the tool is a very important step in determining the features that are required in the builder application itself. In the case of this research project the objective is to bring the ability of 3D interface development into the range of the mainstream user (i.e. non-programmers). In achieving this, one of the key features of the builder tool is its simple high level drag-and-drop style interface development environment. This type of system makes the task of the “3D interface developer” (i.e. the person using the builder tool to develop interfaces) simpler as the tool is more powerful and hides most of the complex underlying detail. While this is beneficial for the “3D interface developer” it makes the development task for the “builder tool developer” (i.e. the person making the builder tool) far more complex. The diagram below gives an idea of where this builder tool fits into the scheme of existing tools and who its target users are.

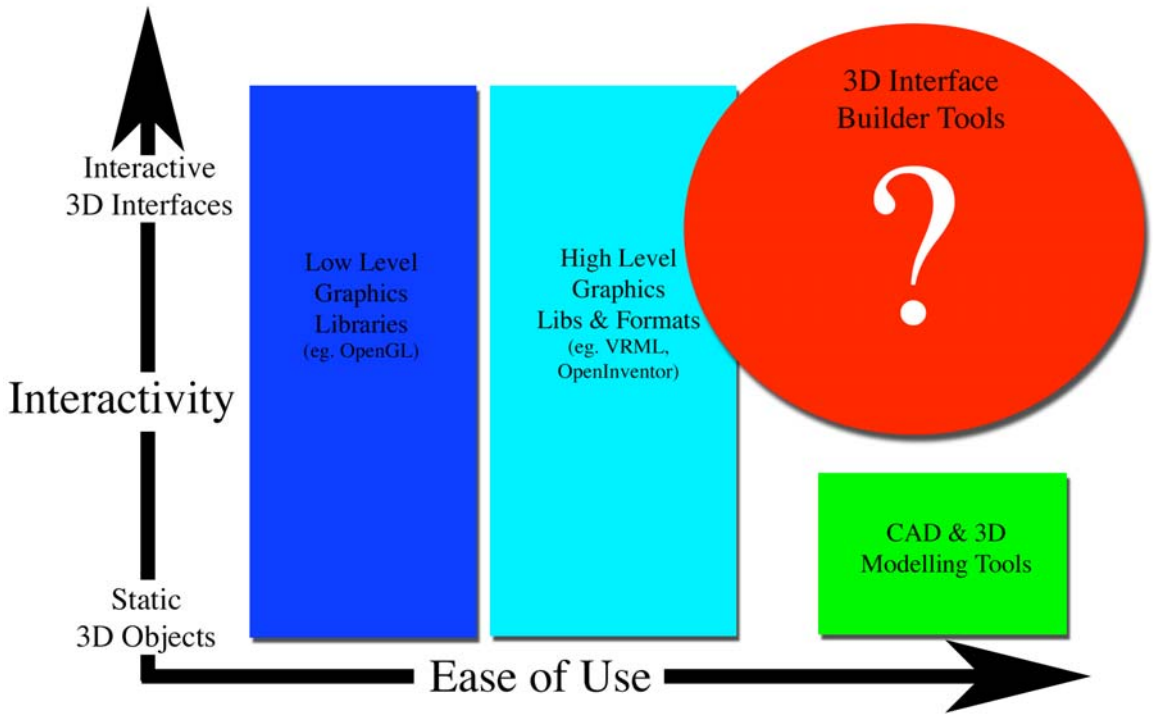



**Figure 5.1: 3D Development tool options**

The builder tool developed in this research project fits into the above diagram in the red section. This tool represents a new level of interface construction environment (i.e. it is the only tool of its type in the red section). When compared to the existing systems that offer the necessary functionality (interactivity), this high level tool represents the first of its kind in the area of 3D user interface development. Other tools such as the low level

programming libraries (eg. OpenGL [Neider 93]), high level programming libraries and extensions (eg. Java3D [Java3D 03], OpenInventor [Hartman 96], Java3DBeans [Dorner 00], [Schonhage 00]) and high level world description formats (eg. VRML [Hartman 96], [Patterson 03a]) all offer the desired functionality but are too low level (i.e. difficult to use, requiring complex programming skills) to make them accessible to a broad developer base. The existing high level 3D authoring tools are either too limited in capabilities (i.e. do not enable interaction at all) or are too low level for the interactive elements (i.e. only enable interaction when it is programmed [Cosmo 98], [Parallel 03]) to enable them to provide the necessary tools to a less experienced developer base.

To achieve the high level development environment as outlined above, the builder must provide a range of features, the key feature being the ability to take a complex programming system (i.e. the 3D interface), hide that detail, and give the developer a "simple to use" method/interface for creating such systems. The features required to provide this high level functionality essentially fall into two key areas.

Firstly the 3D layout environment/tools. This is the physical 3D space into which the interactive 3D content is arranged. The key features needed here are the tools to enable the user to create the visual appearance of the 3D space that makes up the interface (eg. the ability to add, position, orient and size items in the 3D space).

The second key element with regards to providing an effective development platform relates to the need to enable the user to give interactivity to the 3D space (and items in that space) that has been created. It is this second element that represents the biggest challenge. There are numerous existing systems for creating relatively static 3D spaces/environments (eg. CAD, animation and modelling tools), however incorporating a simple to use system of interactions presents a challenging problem. Most of the existing 3D interface development tools/systems simply require the user to have the necessary programming skills to program the interactions into the system. However for this research there is a need to hide this programming detail behind a simpler and more accessible interface.

### *5.2.1 Shaping the 3D Space*

The system this "builder tool" uses for designing and laying out items in the 3D space/environment involves the basic principle of providing an empty 3D space into which items of varying kinds can be added.

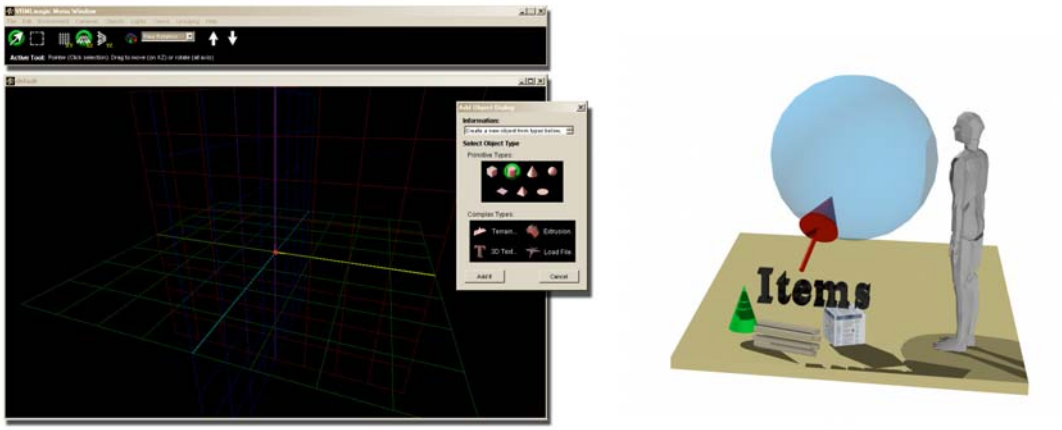



**Figure 5.2: Actual Tool (left) vs. Concept of Tool (right)**

These items, which make up the substance of the world/interface, come from a range of options including:

- **Cameras**

  - as you would imagine cameras are fixed viewing locations, looking at the environment from a particular perspective. Although you cannot limit the users movement in the final world you can use cameras to enable the user to move to known locations or views that are of interest. Cameras are represented in the development environment through an eyeball and target system as shown on left. This system enables the developer to interactively drag the eyeball to a location in space (thus giving you the viewing location i.e. where the user is looking from) and also to interactively drag the viewing target to a location in space (thus giving you the centre of vision, i.e. what the user is looking at).

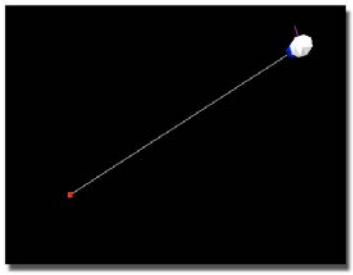


**Figure 5.3: The Eyeball (Camera)**

- **Lights**

  - there are three types of light (i.e. positional or point lights, spot lights and directional lights) available in the builder tool. These lights are represented in the world as shown on the right (lights being the coloured "stars"). In simple terms these items can be moved and rotated in space to generate the desired lighting effect. As lights can have extra attributes such as colour, brightness etc. these additional attributes can be set through a dialog accessible through the applications GUI menu system. By using interactive lighting the effect that the lights have in the space is interactively adjusted based on changes made to the light items in the space (thus providing the developer with immediate feedback on the lighting set-up of the space/environment).

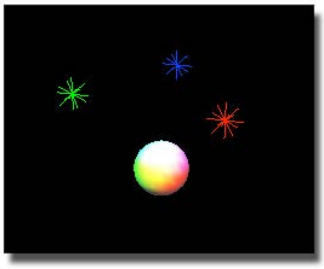


**Figure 5.4: Lights in the Tool**

- **Objects**

  - are essentially geometric structures providing the physical elements that fill the space of the world. As with the other item types, objects can be interactively moved, rotated and scaled within the 3D development environment. In addition

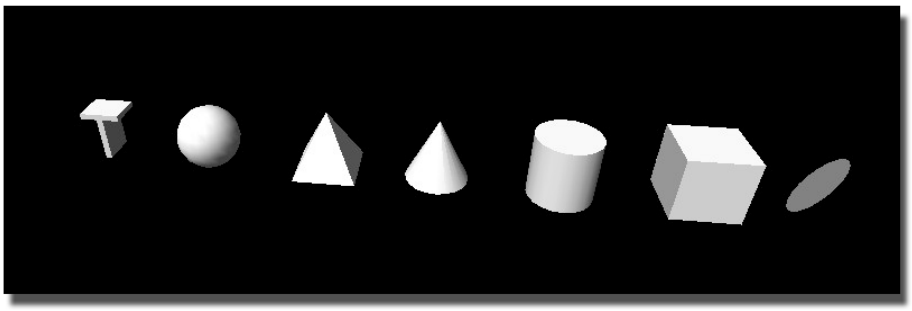


**Figure 5.5: Example Objects**

an objects extra attributes such as colour, textures and other surface and related attributes can be set through dialogs in the GUI based menu system.

- **Pre-defined 3D Components**

  - more complex combinations of other items grouped together to achieve particular tasks. Essentially these are convenience features (i.e. pre-made components for common actions) to enable the quick construction of systems to handle common tasks (eg. the drum interface as shown on right is used for selecting from small sets, much like the 2D GUI menu would be used).

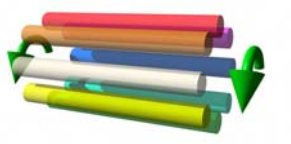


**Figure 5.6: Predefined Components**

- **Global Attributes** (eg. background, fog, …)

  - provide settings that define large effects (eg. background) within the world and although they are not specific items that can be placed in the space of the 3D world, these global attributes can have a significant impact on the appearance of the 3D space/environment.

A collection of these item types can be used to fill the empty 3D space/environment and to create the overall look of the 3D interface.[14] In addition to the basic ability to add these existing pre-built items and components, the system also provides the ability to develop new items/components which can in turn be added to the space/environment. This can be achieved by either building more complex items from simpler pieces (eg. add a sphere to a cone, group them and you have an ice-cream cone, a simple example but it demonstrates how more complex items can be built from primitives[15]), or by simply using more complex geometric models/items, built in other tools and able to be imported into the interface builder application (eg. it is far simpler to use an existing polygonal model of a car than it is to construct a polygonal car from primitive pieces like boxes).

Essentially the builder tool simply provides a graphical system enabling the developer to drag-and-drop physical 3D items representing the various types of interface elements in to the empty space of the 3D world. In addition to this basic drag-and-drop system, the developer is also able to interactively manipulate the items in the space to achieve the desired overall layout (eg. items can be interactively moved, rotated, resized etc. all from within the 3D development environment).

---

[14] Note that the camera and light items as displayed in the development environment are not “real physical” items in the final world, they are simply representations used to enable easier development.

[15] This example uses two simple objects to build a more complex larger object, however it is possible to combine any items into sets (eg. object and light etc).

### *5.2.2 Adding The Interactivity*

The ability to take an essentially passive 3D environment and make it sensitive to user interaction is a critical piece of any 3D user interface. With that in mind it is critical that the system that enables the developer to incorporate such interactivity into the environment be a powerful yet "easy to use" feature of the development tool. To achieve this the builder tool uses an "item centred" system to represent the contents of the world. That is to say the world is an empty space containing a set of items (eg. lights, cameras and objects). Each of those items is a separate entity with its own attributes and interactivity settings. For the purposes of interaction each of these items can be sensitive to user interaction. For example each object has attributes that describe it, such as colour, texture etc. In addition to these "appearance or surface attributes", objects also have "interaction attributes", such as sensitivity to touch.

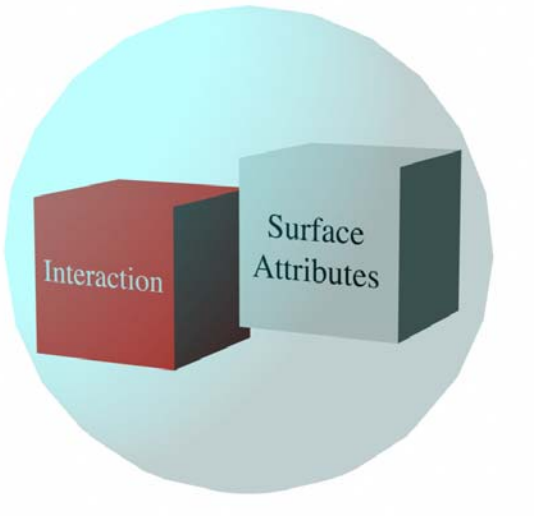



**Figure 5.7a: Example of an object and its attributes**

The world is made up of a set of these items each of which has both structural/appearance information and also interaction attributes associated with it.

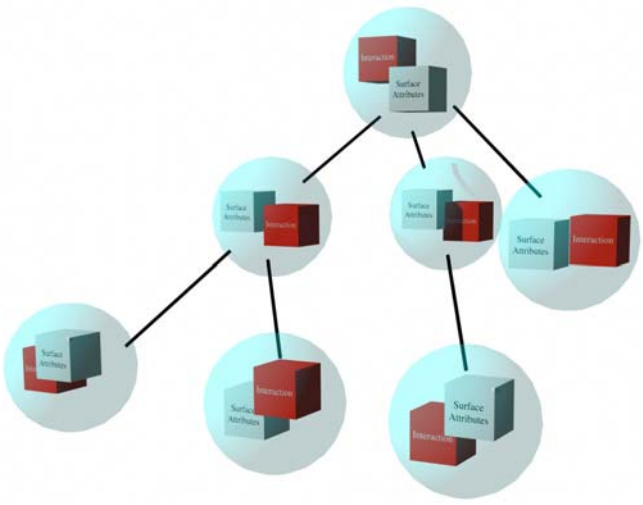


**Figure 5.7b: World made up of tree of items**

Using this object centred approach enables the creation of an unbounded space filled with items that are sensitive to various forms of user interaction. The types of sensitivity supported include:

- **Touch** - touch sensitivity involves being sensitive to user selection (i.e. being clicked on). This sensitivity comes in two forms, firstly it is a sensitivity to selection (i.e. being clicked on) and secondly it is a sensitivity to focus (eg. pointer moved over). When an object is sensitive to these "touches" it can run the relevant action when a user causes either of them to occur.
- **Visibility** - a sensitivity to visibility means that an object is sensitive to being viewed by a user (i.e. when the user can see this item then its visibility action will be run).
- **Proximity** - a sensitivity to proximity means that an object has a sensitive region around its location in space. When a user enters or moves in this region the proximity sensitivity is alerted (eg. a user moves into the range of an object's proximity sensitivity and this triggers the action specified).
- **Drag** - sensitivity to dragging means that an item is sensitive to the user action of selection (eg. clicked down) and then "dragging" (eg. moved while click on). Although this type of dragging is well understood in 2D systems, it also applies in 3D multi axis input systems.

The task of setting these sensitivities is a simple one involving selecting the item in the development tools 3D workspace, then double clicking the item to bring up it's attributes in a dialog box.[16]

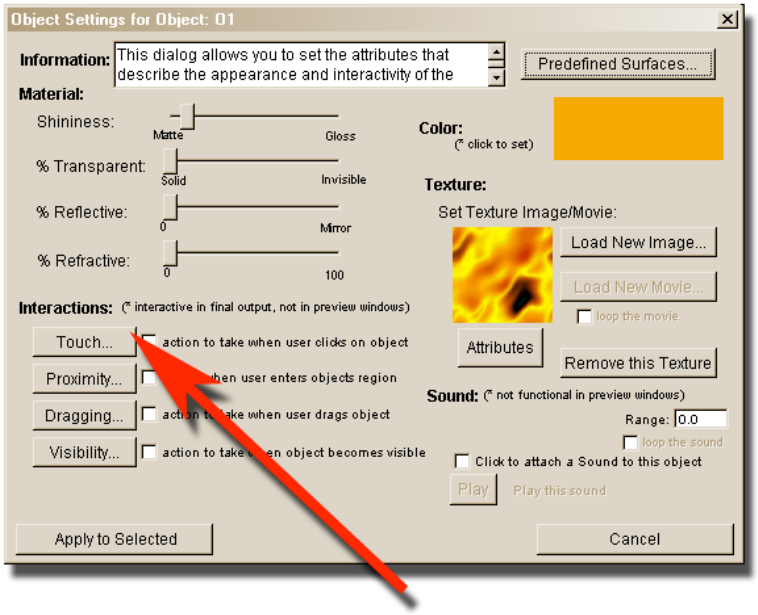



**Figure 5.8: Setting Interaction Attributes**

[16] This can also be achieved through the GUI menu system.

From within the object's attributes dialog as shown above, the user simply clicks the relevant type of sensitivity to move on to the action part of the interactivity control system.

Having made an item sensitive to a type of interaction the next step is to indicate what action to take when that sensitivity is triggered. This "action" stage is far more complex than the sensitivity stage. The primary reason for this complexity is the huge range of possible actions that can be undertaken. Unlike the sensitivities which are limited to a small set of possible interactions, the actions are open to a huge range of possibilities.

Most existing interface development tools handle this "action" element by requiring the developer to hand program the action [Cosmo 98], [Parallel 03]. By doing so the tool simply needs to provide a text editing facility to enable the developer to add the action in the relevant language. Although effective, this system requires the developer to have considerable programming skills, which this research is attempting to avoid, at least in the mainstream. The earliest version of the 3D interface builder developed for this research used this hand program writing based approach (as demonstrated below).

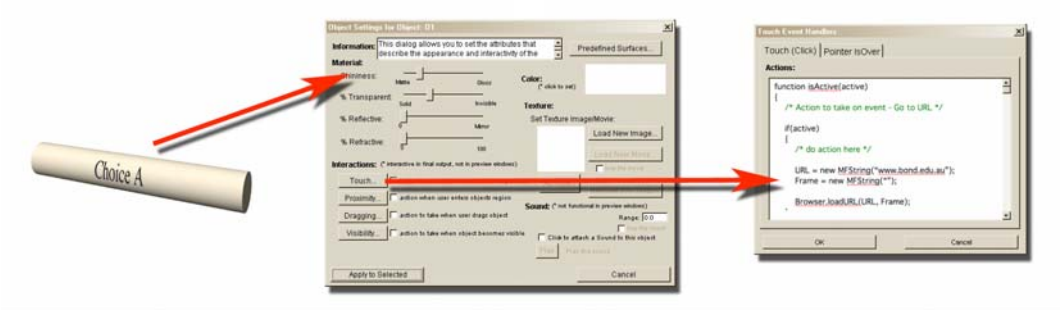



**Figure 5.9: Programmed action settings (i.e. Set Sensitivity, Write Action)**

When used by a developer with strong programming skills this system proved very effective, although somewhat time consuming. However it was quickly discovered that many of the hand written programs were essentially doing simple actions which varied only in the arguments they manipulated (eg. many link to URL actions where the action is identical and only the arguments (i.e. the URL) vary). This clearly identified the potential for an action specifying system that hides the "program" detail and simply allows the developer to choose from existing pre-defined actions and individually set the "arguments" to generate the specific result. The diagram below demonstrates the basic concept for this system.

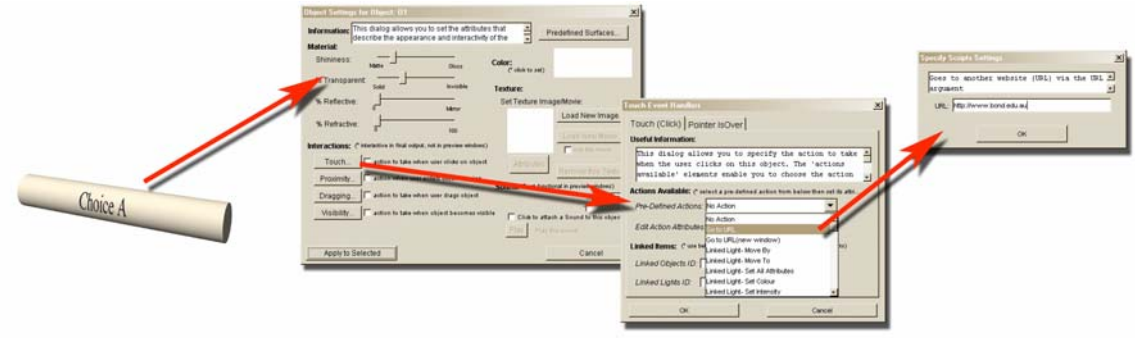



**Figure 5.10: Non programmed action settings (i.e. Set Sensitivity, Choose Action, Set Arguments)**

For many common actions a system of this type is highly effective, both in terms of development speed and accessibility (i.e. no need for developer to need to program the action). Certainly there will be particular complex cases that will require unique, one off programmed solutions, and for these systems the availability of a simple programming interface is maintained. However, in the mainstream, a rich set of pre-defined actions with variable arguments provides the necessary development functionality for many 3D interfaces. Through this pre-defined action/argument setting interface the builder tool enables developers with little or no programming experience to create complex 3D interfaces through a simple development interface.

With this combination of item sensitivity and action setting the user can easily add interactivity to the 3D items that fill his/her world. In doing so a simple passive 3D space can be turned into an interactive 3D interface.

### *5.2.3 Development Platform - Bringing the Features Together*

Bringing together both the world building features and the interaction setting system provides a system for interactively creating a rich 3D interface.

The world construction system enables the adding, positioning and editing of items to shape the environment in which the user moves. The interactivity system then provides the ability for that environment to respond to the user as he/she moves within the resulting world.

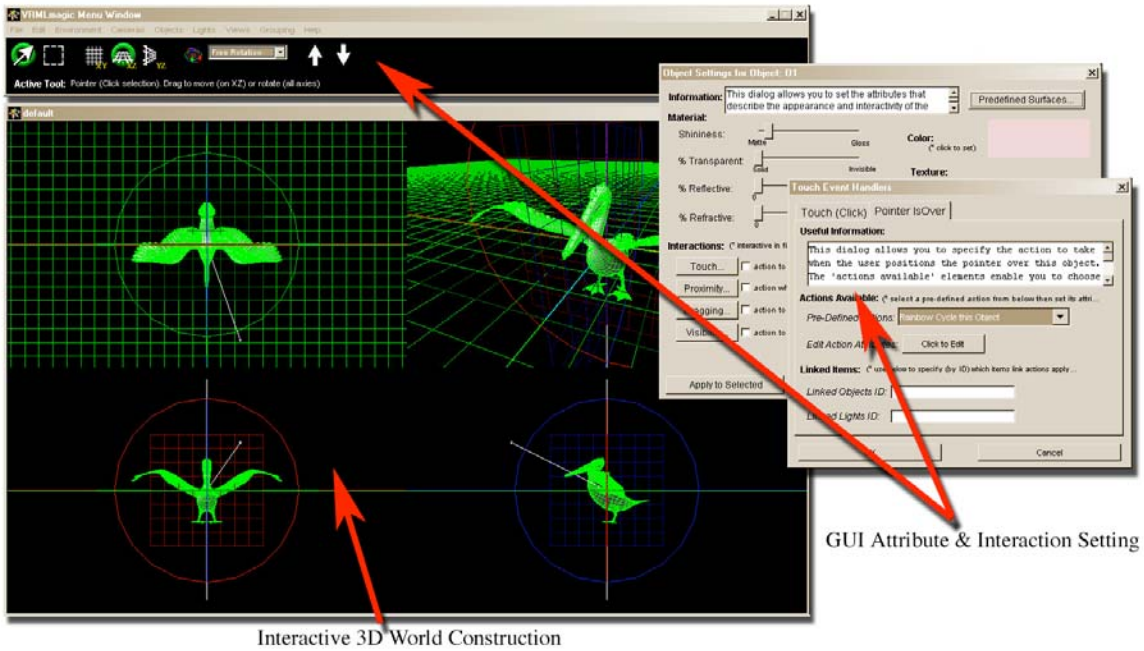



**Figure 5.11: World building features & Interaction control**

From the developers perspective this can all be achieved through a single development tool incorporating the features described above, consisting of both an interactive 3D workspace (i.e. for manipulating items in the 3D space) and a GUI based dialog system (i.e. for setting attributes and actions). With this tool the developer can use simple

interactive settings to create both the physical layout of the 3D environment and the interactivity of the items within that space.

In addition to this basic development system, the builder application also incorporates the ability to create larger more complex items (i.e. “components”) which can be saved as generic working pieces and reused for common interactive tasks. These “components” can then be incorporated into the builder application itself (eg. in the same way as the generic box object is one of the choices in the add item dialog you can put a “component” into the set of basic pieces and thus make it readily available to all the builder tools users).

Bringing this combination of features together makes up the overall builder applications “development environment” and gives the system both a low level (i.e. build components with basic actions) development facility and also a high level (i.e. build an interface by using existing components) development platform.

## 5.3 Implementing the Builder Tool

As described in the above sections the builder tool provides the developer with a full development environment for constructing both the 3D layout of the world and setting its interactivity. This section looks at how that development environment/application is designed and constructed.

### *5.3.1 The Builder Application Itself*

The builder application is implemented as a cross platform executable application. From an external standpoint this application essentially has two key features, the interactive 3D environment for laying out and interactively adjusting the 3D space; and the GUI based menu/dialog system for setting attributes and interactions for items within the 3D space. Although this system sounds relatively simple in a conceptual sense (this is the objective i.e. make the application easy to use and understand), the builder application is actually relatively complex when looking at how it is implemented. Some of the key issues it deals with include:

- Storing the “world” information in a generic form (able to be output/displayed in various formats).
- Displaying in 3D the “world” information.
- Enabling interaction within the 3D environment (i.e. able to interactively drag items in space etc).
- Enabling setting of attributes of the “world” (i.e. able to set attributes unsuited to interactive 3D editing through a GUI or other interface).
- Outputting the “world” information in a suitable format.
- Enabling testing of the resulting “world” from within the development environment.

## Internal Structure of the World

The internal (i.e. within the application program) storage of the information that describes the world is a critical feature both in design terms and implementation terms. From the design perspective the ideal storage system is able to capture all of the descriptive features of the world in a generic (i.e. not tied to any particular format), compact, yet easy to access form. This stored information needs to be flexible enough to be "written out" in one format and "read back in", and in the process be converted back into its generic internal form. To implement this the builder application uses an object oriented "tree of items" structure to store the content of the world.

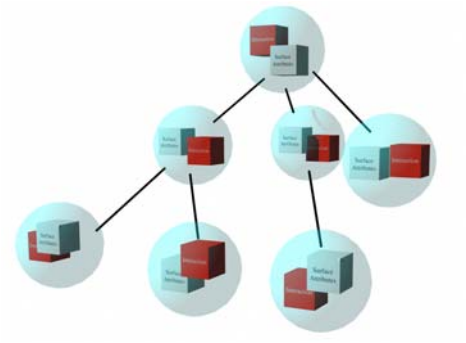


**Figure 5.12: Tree of Items**

Within this tree each branch represents an item in the world (these items can be objects, lights or cameras). Some items, such as the group objects, will have children (i.e. branches with sub-branches). Others will simply be independent items.

Each item in this tree fits into the overall set of possible item classes. This set is based on the generic item and each sub type inherits the basic attributes of its parent.

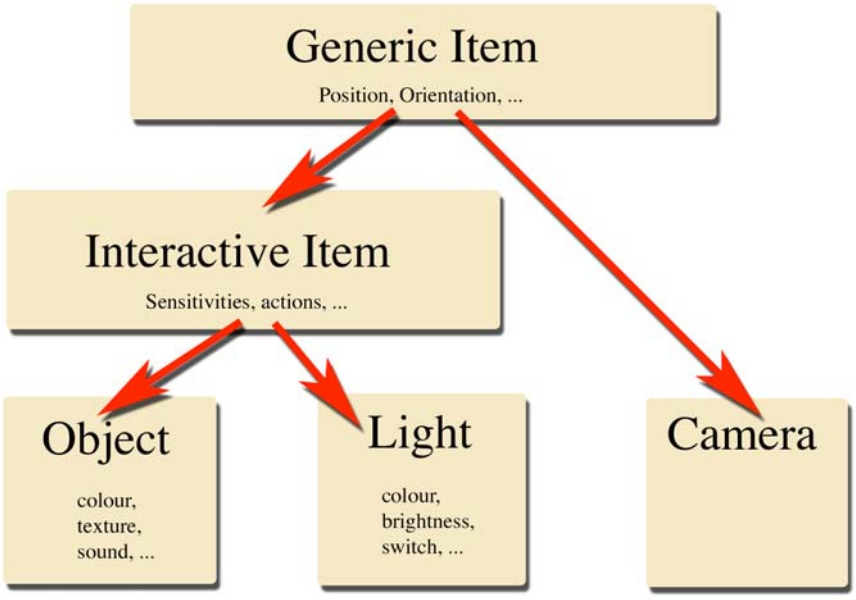



**Figure 5:13: Basic inheritance of item attributes.**

As Figure 5.13 shows each sub class inherits from the generic base class Item. This object oriented approach provides the simplest method for creating a broad range of

complex item classes, each of which builds on the reusable base level attributes. With this set of item classes it is a simple process to implement a set (in the form of a tree) of items that describe a world. It is this set of specific items that provide our internal representation of the world. This representation is independent of any other systems and as such it is possible to output our world in a range of formats, simply by traversing the tree of items and outputting the content in the desired format (note that this same traversal of the tree method is used for the interactive display (using OpenGL) of the world within the builder application itself).

Given this overall layout, it is a simple process to create new instances of various item classes to add new elements into our world. Some of the main classes that fit this "elements of the world" level include:

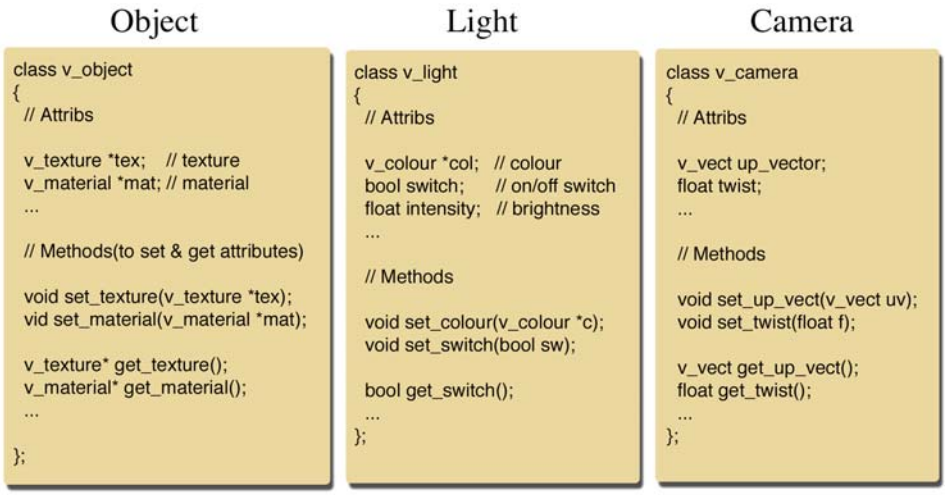



**Figure 5.14: Basic Items and their stored attributes.**

As Figure 5.14 shows these generic classes can simply have their independent attributes set to individualize the item in the world. For example, to make an object sensitive to touch simply requires the setting of the objects touch sensitivity (i.e. by calling the objects internal private method *set_touch_sensitivity(bool touch_state)*). The following code (in C++ as that is how the builder tool stores the internal structure) gives an example of how you might individualize an object and then add it to the tree.

```
…
v_object *RedClick = new v_box();              // create a new instance of a generic box object
RedClick->set_position(5.0, 1.0, 0.0);         // set boxes position in space
RedClick->set_primary_colour(1.0, 0.0, 0.0);   // set boxes colour to be Red
RedClick->set_touch_sensitivity(true);         // make box sensitive to touch
RedClick->set_touch_action("….");              // set action to take on touch

v_object *treetop = world->get_treetop();      // get the top of the existing tree
                                               // if no tree exists this will create one and
                                               // return the top of it
treetop->add_object(RedClick);                 // add RedClick box to tree
…
```

For the “world developers” convenience each of these attributes can be obtained (by the builder tool application) through the user interface of the builder application (i.e. via the GUI systems dialog boxes) and thus the user/“world developer” simply chooses a colour from the interactive colour chooser dialog and this result is passed down and causes the internally stored object to call its “set_primary_colour” method. This same attribute setting method applies across the board to all item types and attributes.

This internal structure is a neat self contained system for representing the contents of a world and can be used to output or display this world using a range of different methods. For the initial version of the builder tool the interactive 3D workspace (i.e. the 3D area in which the user/”world developer” interactively manipulates the content of the world) was implemented using C++/OpenGL to display the content as described in the worlds internal structure. Essentially this system simply traverses the tree of items and draws each item into the 3D space based on the items arguments. See the pseudo-code below for a basic idea of how this works:

```
…
Get top of Object Tree
While (not finished with object set)
{
    get next object in tree
    get objects attributes
    apply objects attributes
    draw objects geometry
}
…
```

For output (i.e. outputting the world to a final interactive form) X3D/VRML was selected as the output format of choice, however a range of other options was considered including Java3D or even C++/OpenGL (although this would have required a separate compilation step in the development process). The method of output is actually very similar to that of the drawing process as described above with the only difference being that instead of drawing the items they are written out in VRML form.

As both the interactive display and the output examples show, the benefit of having this self contained, object-oriented internal structure for the world makes it possible to then apply this “world” in a range of output and display methods.

## Construction of the Builder Application

The key to the builder application is the construction of the pieces that make it work. Essentially it sits above a set of underlying elements that enable it to function. These pieces themselves are extensions of underlying systems.

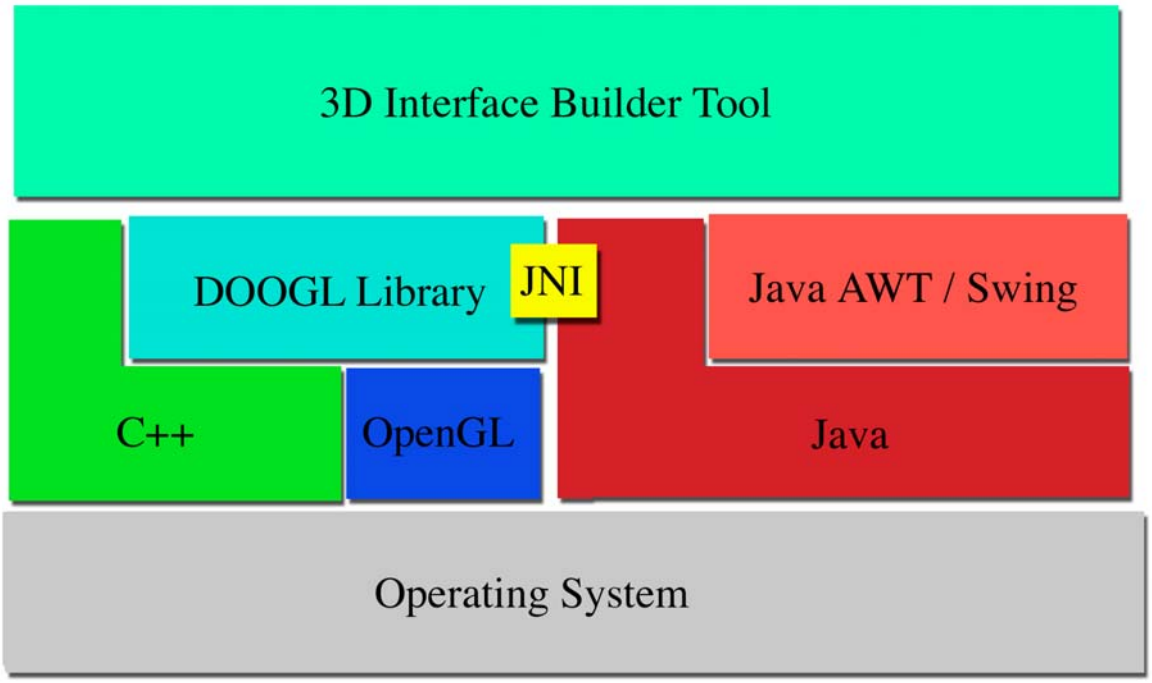



**Figure 5.15: Underlying Systems of the Builder Application**

As Figure 5.15 shows the builder tool makes use of the DOOGL library. This is a library developed as part of this project to provide the basic set of object related functions including information storage, rendering and output facilities. Essentially this library provides the mechanism to control and render the 3D world. In doing so it makes use of the underlying OpenGL and C++ systems to implement its functionality. The primary reason for implementing this library is to reduce the complexity of implementing the 3D world, by making a library, which is essentially a higher level wrapper for the OpenGL functionality. This provides an easier mechanism for manipulating the 3D world and its contents.

In its final executable state the builder application itself is a Java application which makes use of an underlying C++/OpenGL 3D graphics system. Essentially this system works by using the DOOGL (i.e. C++/OpenGL) elements to store the 3D world information and display it to the screen for interactive manipulation; while the Java system provides the GUI interface for setting the other attributes of the world (eg. colour selectors etc). These two systems are linked via a JNI (i.e. Java Native Interface) gateway, through which the GUI system passes down arguments to the C++ world and the C++ world passes up arguments to the GUI system.

Although complex, this system provides the best overall solution in terms of portability (i.e. Java GUI system is cross platform and C++/OpenGL elements can be easily ported) and efficiency (i.e. OpenGL performance is best for interactive 3D work and for this reason was selected).

Several other arrangements (see list below) were considered, however none provided the overall combination as represented in the final builder application:

- **Fully C++/OpenGL** - in terms of interactive performance this is clearly the fastest (with both OpenGL and native windowing libraries both the fastest options). However it is less portable than other systems (i.e. GUI libraries vary from platform to platform) and hence this would have produced a platform specific GUI system.
- **Fully Java/Java3D** - in terms of portability this is clearly the best option with the resulting application able to run cross platform without any porting effort. However the interactive performance of Java3D is somewhat questionable and given that the tool requires significant interactive 3D performance, this is a major limiting factor.
- **Combination System (as implemented)** - although the most complex to develop (i.e. working with multiple languages and the JNI gateway between them), it also provides the best in interactive 3D performance (by accessing the native OpenGL systems) and the portability features of the Java GUI interface.

The builder application uses these technologies to create what appears to be a single application, that provides both an interactive 3D workspace (being controlled by the OpenGL and C++ systems) and a GUI interface (being controlled by the Java systems).

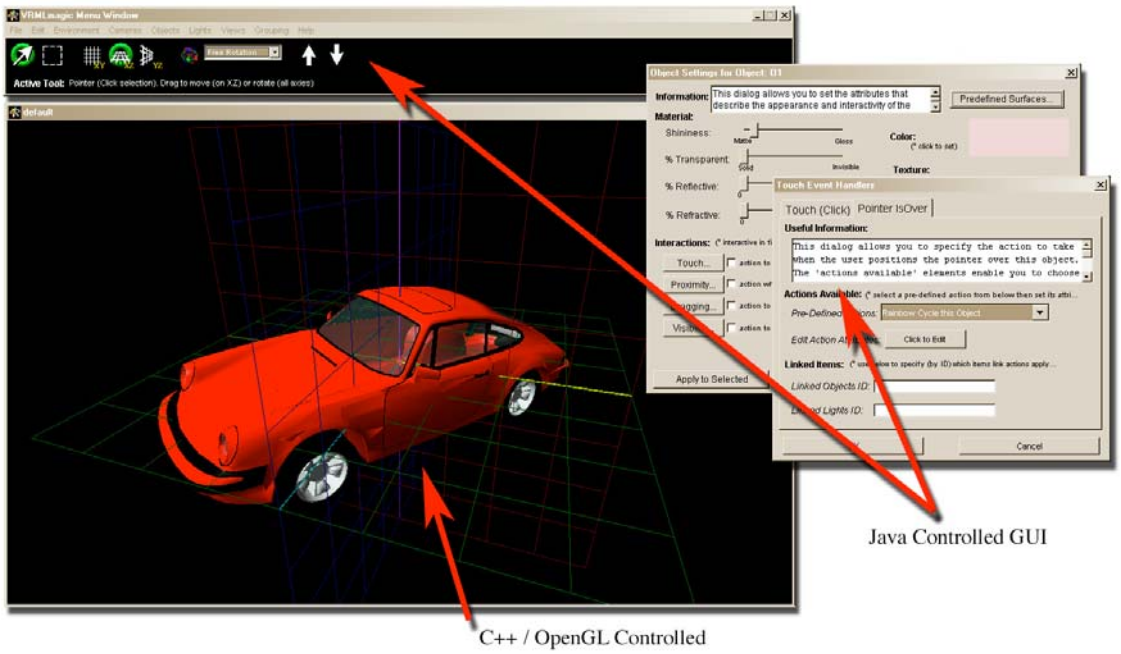



**Figure 5.16: The Builder Application (Java and C++/OpenGL elements)**

### 5.3.2 The Basic Development Environment

The objective of the builder application is to create a development environment in which the user can easily and quickly construct and test interactive 3D interfaces. In achieving this objective the development environment incorporates a number of elements as shown in Figure 5.18:

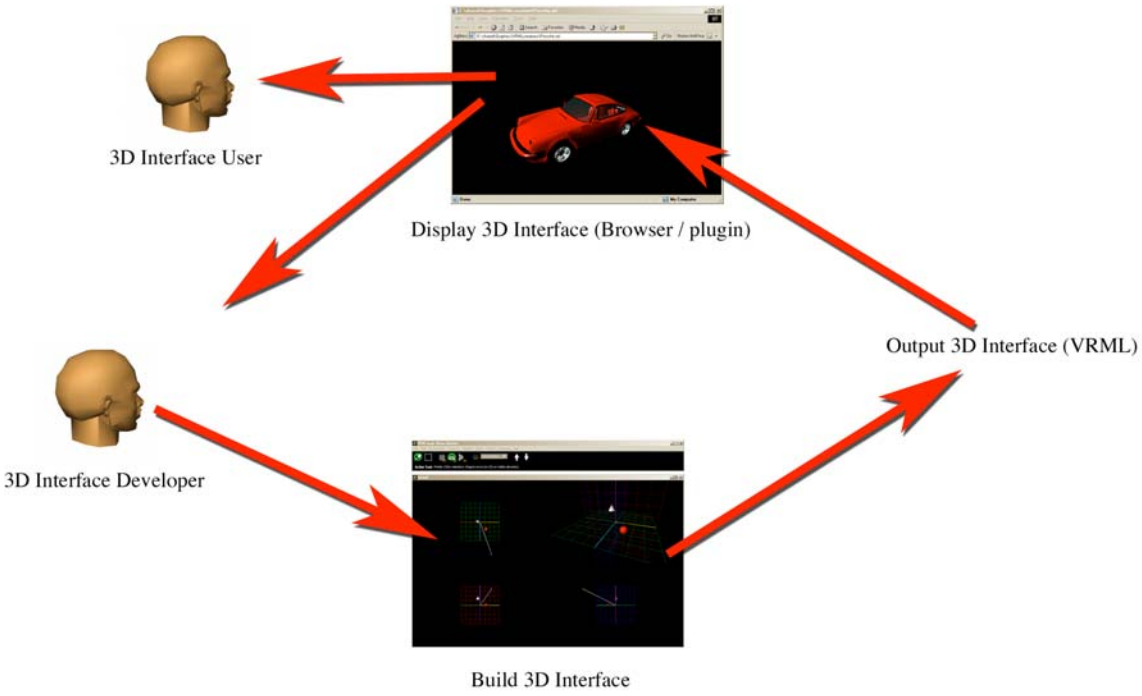



**Figure 5.18: The basic 3D interface development cycle.**

This full development platform incorporates both the builder tool and the final "browser/plug-in" tool. Although these are separate applications, the builder tool can run/preview the resulting system in its final state by simply running the browser tool and thus enabling the developer to preview his/her 3D world without stepping outside the development system. From the "user/world developer's" perspective the builder tool provides a platform in which he/she can interactively build a world (using the object centred model and the set existing actions, components and item types) and then test that world (in its final form) using a browser/plug-in application. This overall system describes the basic design for the workflow pattern for a 3D interface developer.

The diagram above (Figure 5.18) demonstrates the workflow for the current (i.e. VRML output) version of the builder tool, however given the tools ability to output to various formats a more general workflow could be described as shown in Figure 5.19.

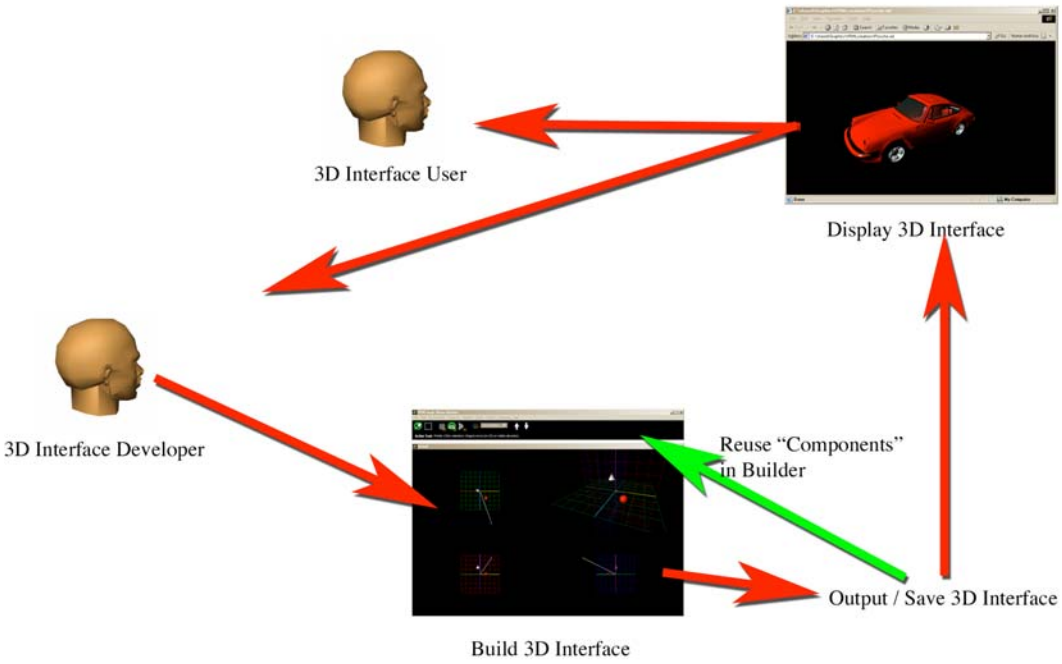



**Figure 5.19: Generic Development cycle.**

As this more general workflow demonstrates, the developer can either simply use the builder to construct interfaces or can use the builder to build additional components which in turn extend the capabilities of the builder tool for future applications.

This extendibility feature of the system is a powerful mechanism for enabling the development of new interface components and interactive actions, thus enabling future users to build larger and more complex systems.

This entire development platform is essentially based on the builder tools ability to store and manipulate the "world" and its contents. By hiding the complex detail (eg. the tool handles the detail of how the items are stored, displayed and how their attributes are set and output) the builder tool provides a system that enables the user to focus on the higher level interface development issues.

## The Component Set

The concept of the component set is a key feature to making the overall development system work. Essentially it is a simple concept enabling users to build larger items (or components) from smaller pieces. Thus creating new "components" which themselves can be used in future projects. This is a particularly powerful method for enabling the development of 3D interfaces as it allows systems to build on each other (making use of existing systems rather than reinventing the wheel each time).

A component is basically an item in the world that has a particular set of features, attributes and interactions. Often these items are complex and contain many sub-items. In many cases the type of interaction a component was designed to handle can be applied to other tasks and as such the component becomes a reusable interface element. When

simply saved as an independent item from within the builder tool this “component” can then be reused in various applications and potentially incorporated into the builder tools base set.

Essentially these “components” function as items that sit outside the basic builder application (see Figure 5.20).

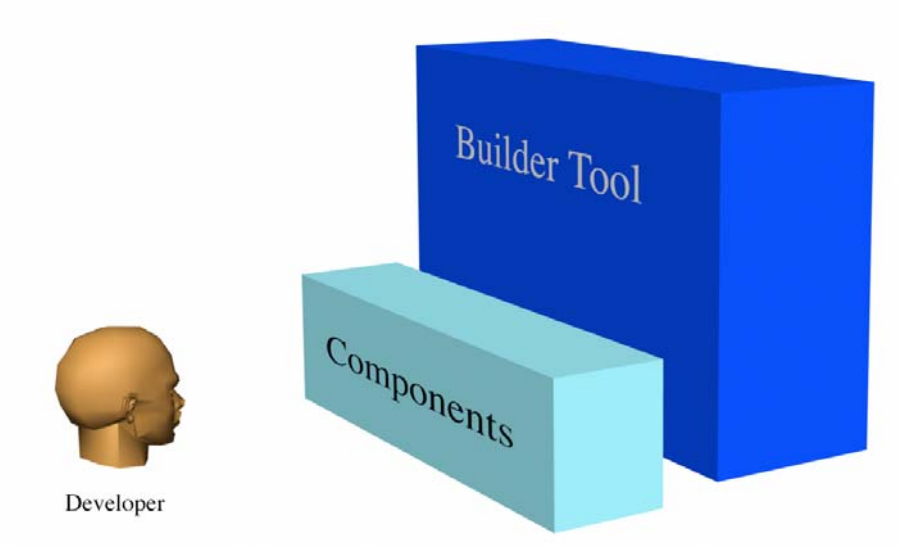



**Figure 5.20: Components and their relationship to the builder.**

As the diagram demonstrates the components exist at a layer above the base builder tool and as such are independent items which can either be used or not used as desired by a developer. In fact a component can be taken and altered to suit a particular purpose. This method provides developers with the option of using existing “pre-built” components to handle particular types of interaction or to develop new methods as desired. The builder itself does not enforce any particular usage policies and as such the developer can make use of these component facilities (i.e. build new components, alter existing ones etc.) as he/she desires.

## 5.4 How it Works: Using the Construction Tool

The previous sections have described the builder tool in terns of its features and how they are implemented. This section looks at the tool from the users (i.e. a 3D interface developers) perspective. From the users perspective the builder tool provides an empty space, a set of available components and a set of basic items and actions with which a 3D

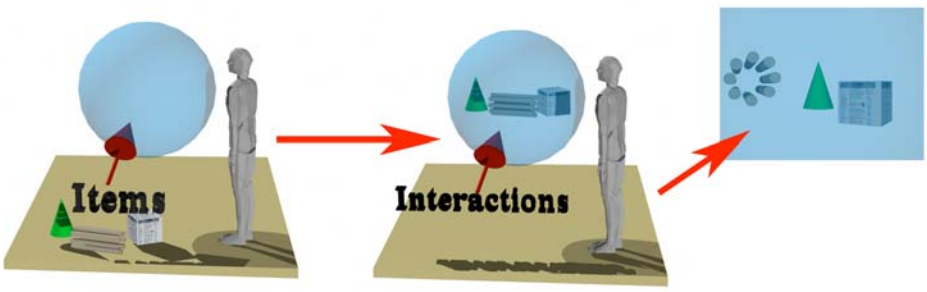



**Figure 5.21: Insert Items into Space, Add interactivity to items, Preview Final Interface**

interface can be constructed. Essentially the user takes these pieces and puts them together into an effective interactive system.

As Figure 5.21 demonstrates, the user constructs his/her interface by adding items, making them sensitive to interaction, then attaching actions to those sensitivities. Thus making the items responsive to user interactions. Having constructed the interface the user then tests the system by previewing it in its final form. All of these steps are smoothly handled from within the builder applications development environment.

### *5.4.1 Example Application of the 3D Interface Builder*

This section outlines an example use of the 3D builder application. In this case the example is the construction of the drum component. This "drum component" is one of the more complex components and incorporates several interactive elements (and as such is a good example for demonstrating the way a user would make use of the builder application). The key purpose of the drum component is to provide a 3D interactive method for selecting an item from a small to medium sized set of choices.[17] This section will go through the step by step process a user of the builder application would take in developing such a component, and through this demonstrate how the builder tool works and also how effective it is in enabling the development of interactive 3D interfaces.

#### Getting Started on the Drum Component

When starting on the development of a 3D interface, the first issue to consider is the general concept of how the interface will visually appear. This conceptual description needs to include both its static layout and also how that layout changes with user interaction. In this example the idea is to have a set of items (i.e. the choices) represented by simple cylindrical items. Each of these items represents a unique choice.


**Figure 5.22: Single cylindrical item**

[17] For a more thorough description of this component and how it works see the Drum Component section later in the thesis.

The set of these individual items will then be arranged in a simple cylindrical pattern as shown in Figure 5.23 below.

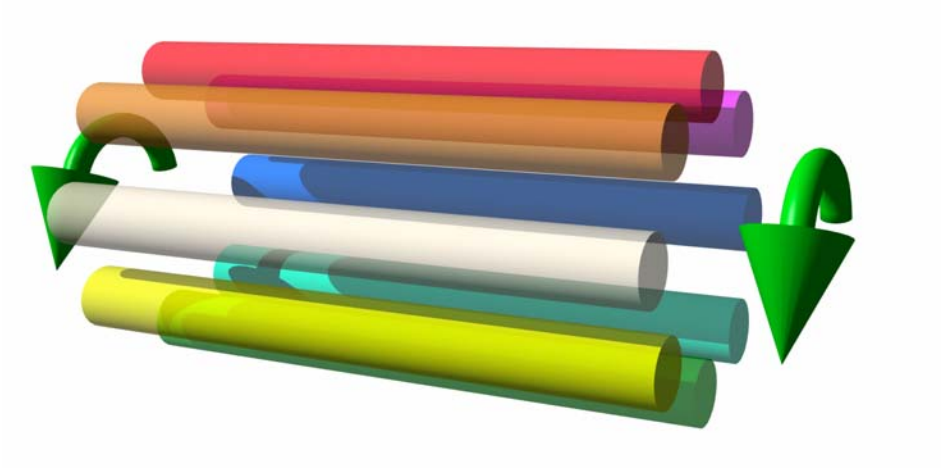


**Figure 5.23: Concept of spinning cylinder of items**

The final part of the component will involve the ability to spin the set (when user specifies) to view the differing items. With this basic conceptual design in mind the developer can then begin building the functional component. The first step is to run the builder application. At start-up the builder application presents the user with an empty 3D space (as shown in Figure 5.24).

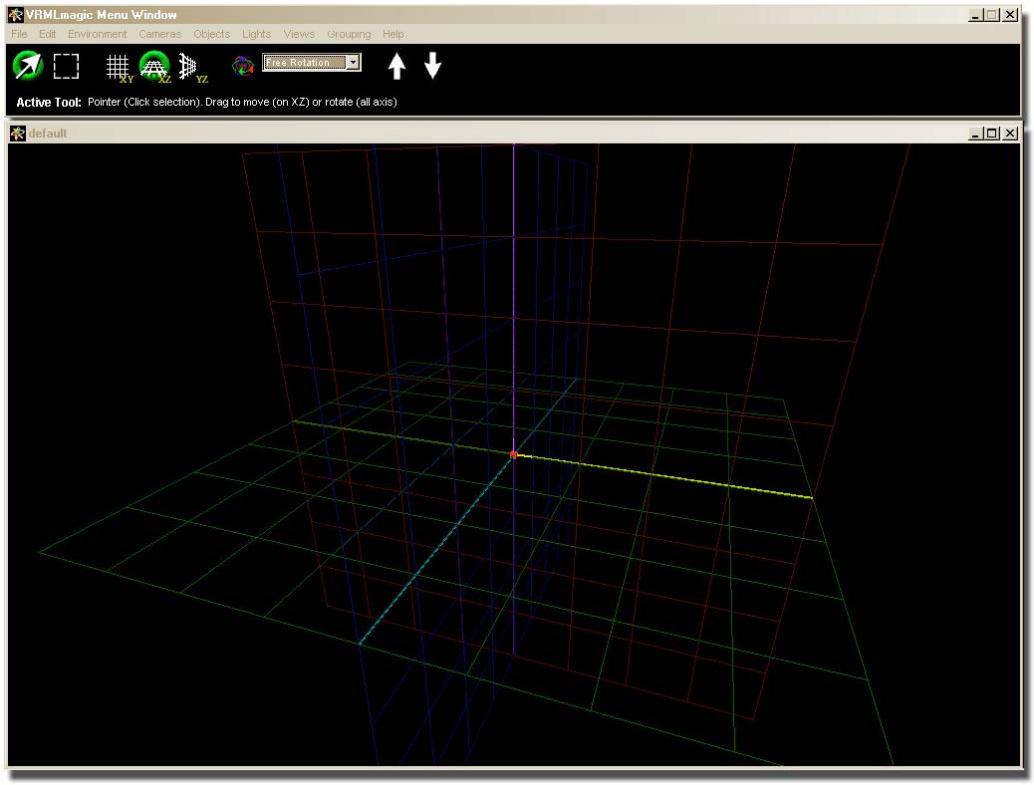



**Figure 5.24: Empty World (Builder tool screenshot)**

## Building the Drum Item

The first step is to construct a basic cylindrical item. This generic item can then be reused as the base for each specific option in the final set. To achieve this from within the builder application do the following:

- Using the menus select *Add Object*
- Choose a Cylinder Object from the dialog (as shown on right, Figure 5.25)
- Resize the cylinder to be long and thin (using the resize menu item and dialog)
- Set the Transparency to be ~30% (to enable user to see through it a little)
- Set its colour and texture as desired (as this is generic leave as defaults)

This will give a world as shown below (Figure 5.26), consisting of a simple elongated cylinder located centrally in the overall space.

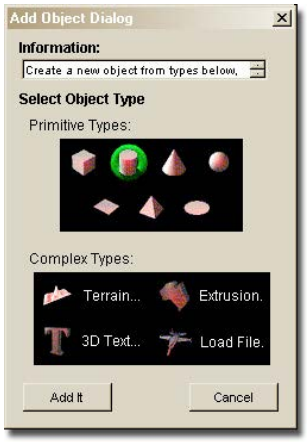



**Figure 5.25: The Add Object Dialog**

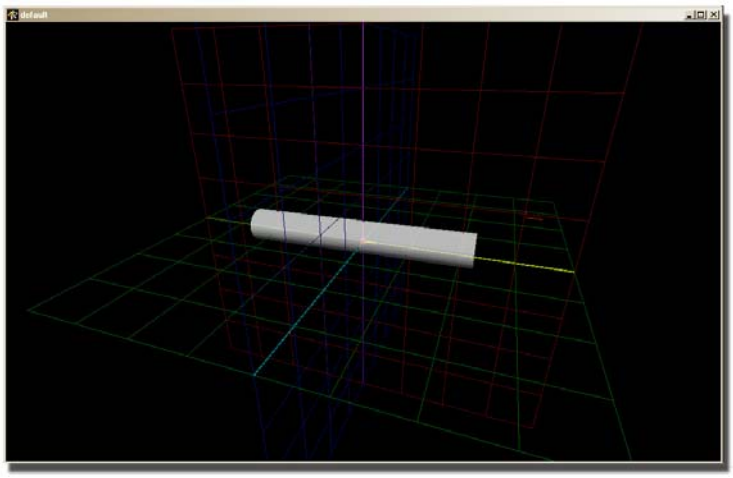


**Figure 5.26: The basic DrumItem**

At this point there has been nothing particularly difficult (or unique) about this process. In fact you could achieve the same result with any one of a handful of existing CAD, animation or other 3D applications.

From this point on the development begins using the features that are unique to this builder application. Starting with applying a simple interaction to the cylinder.

- Bring up the Edit Cylinders Attributes dialog (as shown on left in Figure 5.28 below)

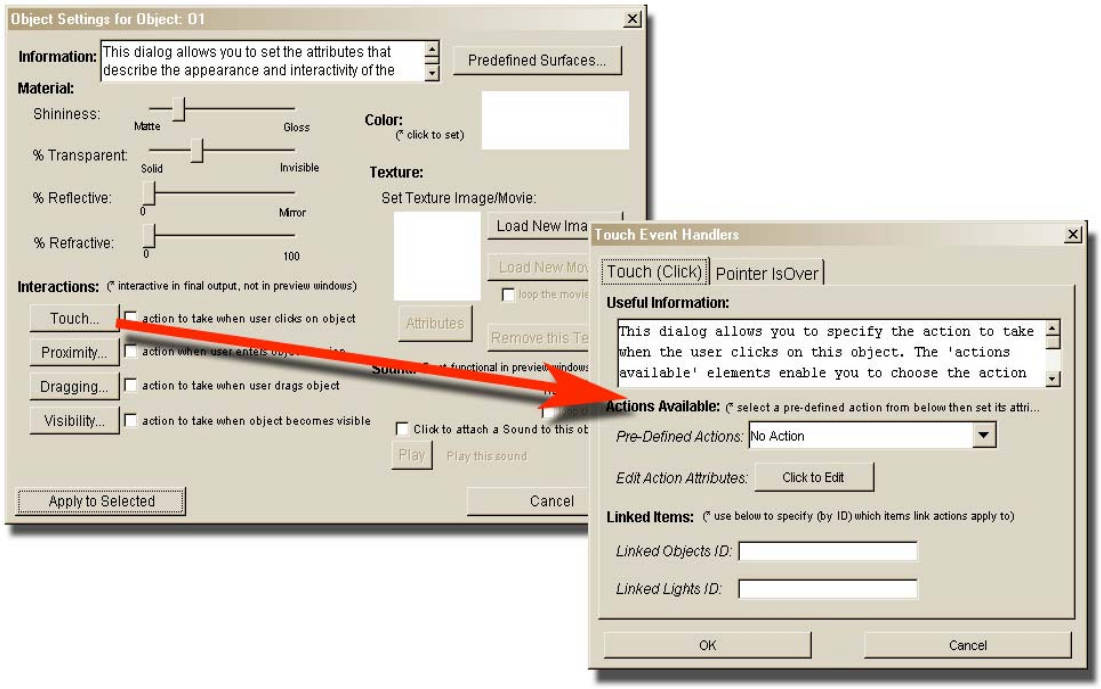



**Figure 5.28: Attribute Setting Dialog (left) with Touch Event Handler dialog (right)**

- Click the Add Touch Interaction button (brings up the touch interaction dialog as shown in Figure 5.28 on the right)
- Choose the "Touch (Click)" pane (this represents actions to take when the user clicks on the item)

  From the menu choose the action to take when clicked on (eg. Go to URL)

  This brings up a dialog requesting the URL information (enter that as required)

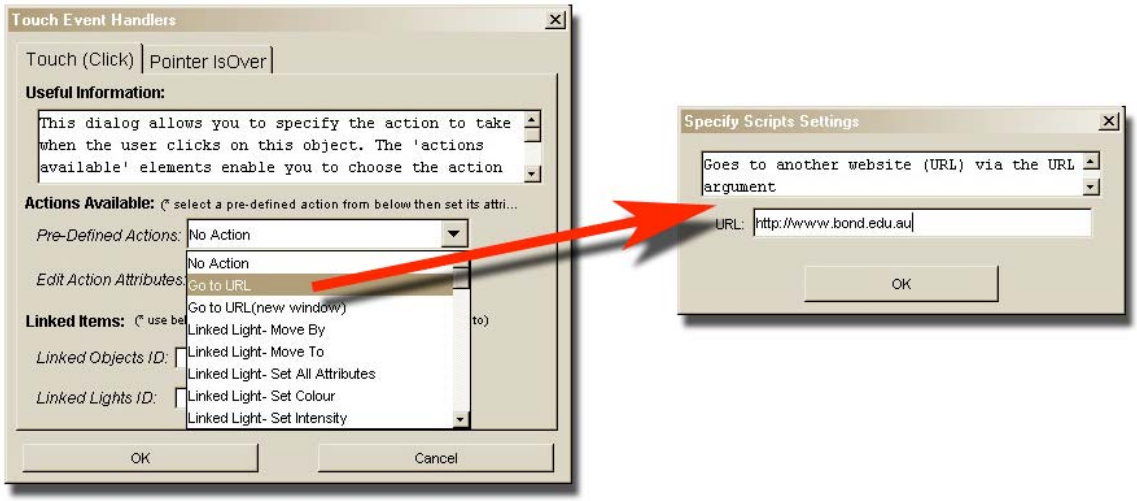



**Figure 5.29: Select action from popup menu then set attributes**

- Now choose the “Pointer IsOver” pane (represents actions to take when the user moves the pointer over this item)

  From the menu choose the action to take when IsOver is true (eg. Resize while Over)

  This brings up a dialog requesting the size amount to enlarge (enter 2.0 to double the size when user moves over it).

- OK the Touch interactions dialog
- OK the Attributes Dialog
- Save the world as DrumItem

This saved item is the generic DrumItem (representing an element within the larger Drum component). This is in itself a complete simple component which swells to twice its size when moved over and transfers the user to a new URL when clicked on. As demonstrated above, the DrumItem, although relatively simple, was able to be created without the need for any programming or scripting and in fact only required the user to choose from menus of interaction options and then specify the specific attributes (eg. the size to enlarge by or URL to go to). With the basic DrumItem complete the developer can now move on to constructing the more complex Drum itself.

## Building the Drum

The first step in building the Drum itself is to create a new world from within the builder. This takes the developer back to the initial starting set-up as shown in Figure 5.24 above). The next stage of the development involves the careful placement of items in the 3D space (i.e. to arrange the set as desired), and as such this is a good opportunity to go into the four view mode for interacting with the world. Four view mode simply presents the viewing window as four separate sections (i.e. quarters) including a top view, side view, front view and a perspective camera view (as shown in Figure 5.30 below).

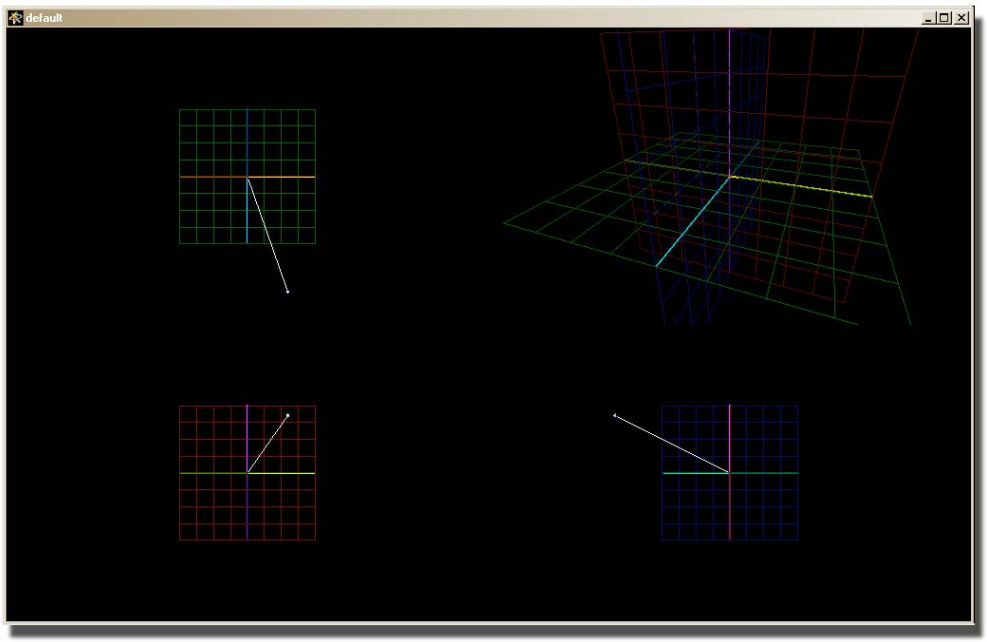



**Figure 5.30: Four view window.**

With this basic set-up in place begin by implementing the steps below to build the Drum component

- **Make the Set of Options:**

  Using the menus select *Add Object* and choose the *Load File…* option.[18]

  Select the DrumItem file and Click OK.

  This will load a DrumItem component (which will now appear in the world).

  Select the DrumItem Object and Clone it N times (i.e. *Edit - Clone)*, where N is the number of options desired).

  You will now have N DrumItems all sitting on top of each other in space.

- Arrange the Set:

  In the Side View (i.e. the blue YZ plane in image below) drag the individual DrumItems into position to achieve a circular arrangement (as shown in Figure 5.31 below). Note that using the side view makes this much simpler as you can clearly see the shape you are seeking to achieve.

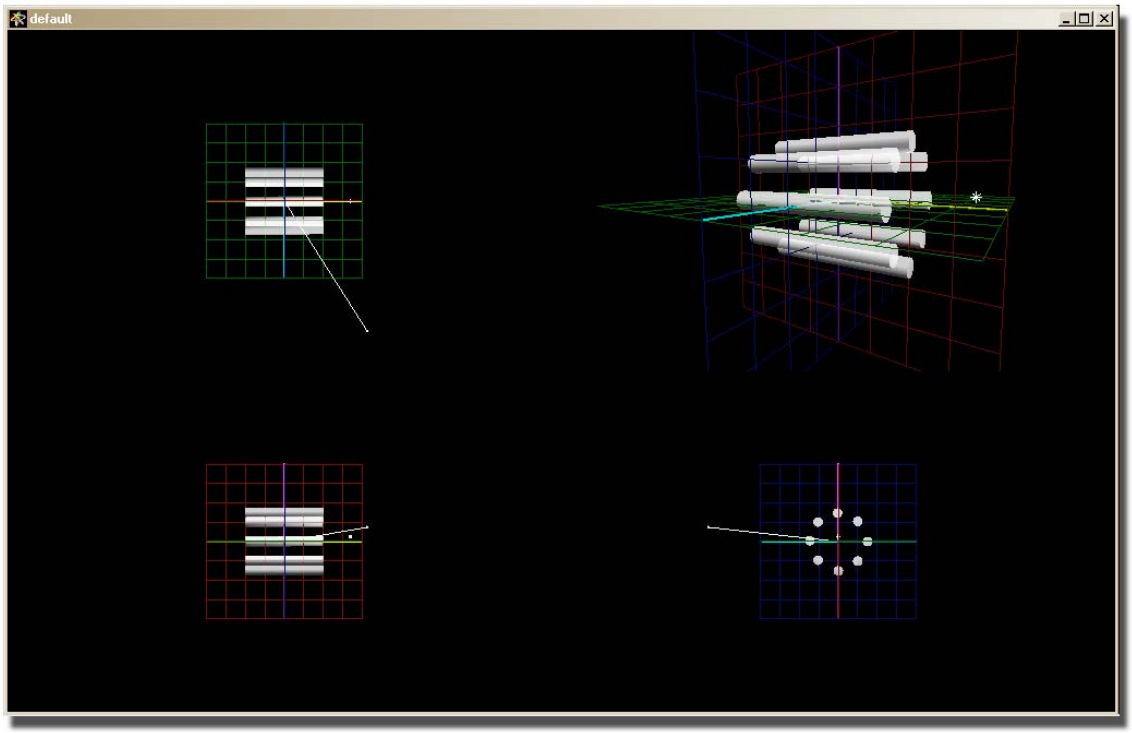



**Figure 5.31: DrumItem cylinders arranged in a cylindrical pattern.**

[18] In the future common components (eg. the Drum) will be added as Object types and will be able to be directly selected (eg. like the terrain or 3d text currently are) from the Add Object dialog.

Select all of the DrumItems (shift click them).

Make a group of this set (use menu *Grouping - Permanently Group Selected Objects*).

Bring up the Attributes dialog for the Group object

Note the group objects ID (i.e. *GroupsID*) as this will be required later

- **Personalize the Set (optional) -** this is the task that a developer would undertake to specify each DrumItems individual action and appearance for a specific implementation. In this case simply set the colours to be different to make it easier to identify the individual DrumItems).

  Select each DrumItem and set its colour (use Attributes dialog as before to achieve this).

- **Create the Initial Spin Controller**

  Using the menus select *Add Object* and choose the *Load File…* option.

  Select the SpinController object file and Click OK (this is a simple geometric object made up of a pair of cones and a torus (i.e. ring) as shown on right (Figure 5.32). Any object could be used for this task, perhaps a cone (or set of cones) is the best of the primitives).

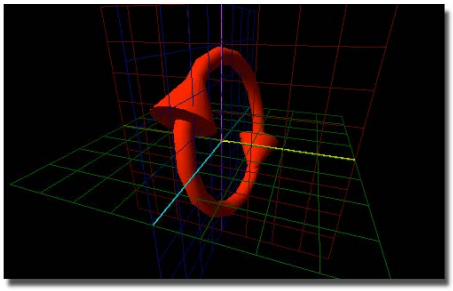
  

  **Figure 5.32: Spin Control Device**

  The SpinController will appear in the world.

  Bring up this SpinControllers Attributes dialog

  Click the Add Visibility Interaction button

  - From the menu choose the action to take when SpinController is visible (eg. Spin about X Axis)
  - This brings up a dialog requesting the rotation speed (enter that as required)
  - This will cause the SpinController to be constantly spinning while it is in view. Although not necessary this is a good mechanism for attracting the users attention to the spinner device and thus enabling him/her to quickly take control of the component.
  - OK the Add Visibility dialog

  Click the Add Touch Interaction (this brings up the touch interaction dialog)

  - Now Choose the IsOver pane (this represents actions to take when user moves the pointer over item)
  - From the menu choose the action to take when IsOver is true (eg. Spin Linked Object about X Axis)

- This brings up a dialog requesting the rotation speed (enter value as desired).
- Set the Linked Objects ID to be equal to the groups ID as recorded earlier (i.e. *GroupsID).* This will spin the set of items (as identified by *GroupsID*) when the user moves his pointer over the SpinController device.
- OK the Touch interactions dialog

OK the Attributes Dialog

The spin controller is now a functional item which sits in space spinning in place. When the user moves over it that causes the set of DrumItems to rotate in place thus cycling the back items to the front and vice-versa.

- **Make the second Spin Controller**

  Select the SpinController Object and Clone it N times (i.e. *Edit - Clone)*, where N is the number of options desired, one in this case).

  You will now have two SpinControllers each of which will function in controlling the spin of the set.

  Move the Spin controllers to the desired locations on either side of the set.

- **Save the final DrumComponent**

These steps complete the construction of the Drum component and this generic component can now be used in a range of specific cases. In order to implement a specific example a developer would simply load the generic component as built above. Then set the individual DrumItems attributes (i.e. give each a new appearance and action as appropriate for their task).

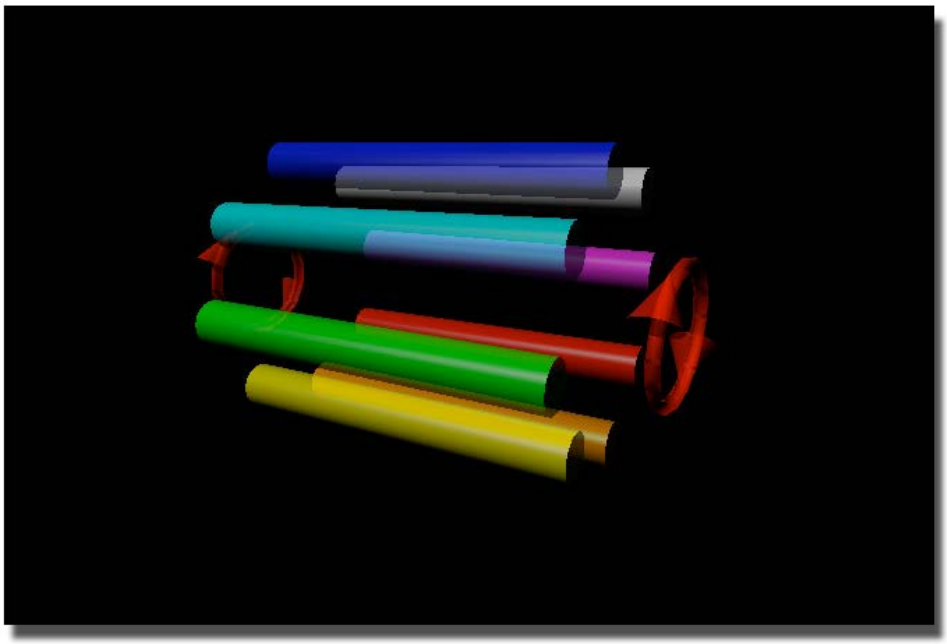


**Figure 5.33: Screenshot of the Completed Interactive Drum Component.**

The entire construction system (as implemented by the builder tool) is designed to be simple to use and to avoid the requirements for users to have specialized skills (eg. programming ability). As the above example demonstrates a user need only have a basic understanding of three dimensional space and basic interaction systems to use this tool to develop quite complex 3D interactive systems.

This example demonstrates the development of a single piece of a larger system (the drum component). The more common use of the builder will be to take the larger components (like the drum itself) and put those together into interactive systems (in the same manner that the example took the simple DrumItem and put multiple versions together to make the drum we can put many drums together to make a more complex selection system). This example was chosen as it enables the demonstration of both the ability to build interactive systems from simple base components (eg. cylinders and other primitives) and also demonstrates the potential for building the larger scale systems from the components already developed (i.e. the DrumItem).

This ability to develop complex systems (with only a limited understanding of 3D and interaction systems) by using a powerful construction tool and building on already developed pieces (i.e. existing components) provides many new users the opportunity to develop 3D interfaces.

As the example demonstrates, the builder application can effectively hide much of the complex 3D graphics from the user and allow him/her to focus on the broader 3D interface design issues. Not only does this allow experienced users to focus at a higher level, but it also brings a whole new set of developers into the field of 3D interface design (i.e. developers for whom 3D interface development is currently beyond their range of skills, but with a tool like the builder application this opens up the possibility of creating 3D interfaces).

## 5.5 Discussion

Throughout this section the difference between users and developers has become less clear. By enabling less experienced users to become involved in the 3D interface development process the difference between developer and user has blurred somewhat. This is primarily due to the perspective from which we are viewing the builder application. In essence this application can be viewed from one of three perspectives:

- **The Builder Tool Developer**: the person who develops the builder application by writing the necessary C++ and Java code.
- **The 3D Interface Developer**: the person who uses the builder tool to construct 3D interfaces.
- **The 3D Interface User**: the person who uses the 3D interfaces as output by the builder tool.

The difficulty that arises involves the fact that the “3D Interface Developer” is capable of constructing “new components” and thus extending the builder tool itself. The question that arises from this is does this make him/her a “Builder Tool Developer”? For the

purposes of clarifying the roles it does not; however in the real world this empowering of the 3D interface developer is one of the biggest contributions of this application.

Another boundary that is blurred through the use of the builder tool is that of the distinction between the builder tool, the resulting 3D interface and the browser or display tool in which the interface is displayed. Essentially the builder tool is simply a mechanism for constructing a 3D interface. That interface is then output in a format which can be used by a display system (i.e. in this case the display system is a browser application using a VRML plug-in). This distinction between viewing the final world (i.e. in the browser) and building the world (i.e. in the builder) is an important one to understand.

### *5.5.1 Choosing the Outputs*

One of the key development choices in this research involved the choice of VRML as the output format. This decision is based on a number of comparative benefits that VRML offered over other options. Essentially VRML was the most suitable output format for the specific task at hand, however by using the generic internal structure for storing the world and its state (i.e. within the builder tool) the attachment to VRML is simply one of convenience. That is to say that other output options are possible and would require only limited changes to the system (i.e. new output methods for item types) to implement. VRML was selected because it offered the desired interactivity features as well as networking capabilities which were ideally suited to many of the particular tasks that the components were designed to handle (eg. search results).

### *5.5.2 Options for Interface Development*

When looking at the options that are available for 3D user interface development it is clear that there is a need for more high end tools. Tools that are not unlike the builder tool described in this section. The risk that comes with these high level tools is that they may stifle the innovation of developers. That is to say that once developers can use an existing component to achieve a 3D task, they will simply use that component rather than looking at developing new and innovative solutions. This has its positives and its negatives; in a way providing a single reusable solution guarantees that interfaces will have at least a base level of functional features and brings a level of consistency to 3D interfaces (a desirable feature for any user interface). However it may reduce the number of new systems that are developed to handle that particular task. Although this is true it is also true that tools, like the builder tool described, also make the development of new components an easier task and as such users will have less difficulty constructing their own “components” than had previously been the case. This potentially leads to an explosion of new and innovative components for a range of specific tasks.

On the topic of interface development tools, it is certainly true that other development tools do exist, although they are targeted at a lower level (i.e. programming required) audience. Tools such as OpenGL/OpenInventor provide powerful programming mechanisms for developing 3D interfaces. The difficulty is that they are difficult to program with, and even for programmers (with the necessary skills), these programming tools take a significant amount of development time to use in building a 3D interface. At

a slightly higher level (than these programming languages/libraries) are tools like VRML itself, certainly simpler to use than programming libraries, these formats allow users to specify worlds at a much higher almost descriptive level. Although these high level formats provide an easier solution they still require a relatively complex understanding of the format and a related scripting or programming language to be used in implementing a complex interactive system. Unfortunately it is the interaction that introduces the complexity to these systems (i.e. VRML worlds are quite simple to write if you do not incorporate interactivity), but without interaction you cannot have a 3D interface. In simple terms the builder tool fits a gap in the available development options for 3D interface builders. By providing an easy to use development environment the builder tool enables developers to work at a higher level of interface development than would be possible with other existing tools and in the process provides a mechanism for developing new an innovative 3D components.

## 5.6 Summary

The builder tool offers a platform from which developers can create complex 3D interfaces through the use of a simple interactive interface. This tool, which is based on a similar concept to that implemented so effectively in 2D GUI interface development tools, enables users to simply drag and drop 3D components into a 3D environment to create interactive 3D interfaces. These components can then be manipulated to achieve the desired visual and interactive appearance and results. By providing an object centred interaction system involving the ability to make items sensitive to interactions of various types and then attaching actions to those sensitivities it is a relatively simple task for developers to introduce interactivity to their 3D environments. Previous research has identified the lack of development tools as a key area holding back 3D user interfaces.

> ‘ There is a scarcity of effective, practical Web3D applications. There appears to be several reasons for this phenomenon. First, there is a general lack of understanding among developers of how to use and integrate 3D graphics effectively for various problem domains. In other words, it is unclear as to what types of tasks and users are best served by presentations that incorporate 3D graphics with other representational forms. Secondly, integrating 3D graphics into presentations is potentially time and resource intensive, due to the limited availability of integrated authoring tools.’ [Zimmerman 03].

The builder tool described and constructed in this research contributes to solving this problem by demonstrating an effective high level 3D user interface construction tool (the first of its kind). This tool enables a set of less skilled developers to, for the first time, enter the field of 3D user interface development. By also creating a set of new “3D components”, each designed to solve particular real-world tasks, this tool also contributes to addressing the first issue raised by Zimmerman above (i.e. developers can use existing components to handle known types of tasks thus clarifying how and where 3D interfaces can be effectively applied).

The builder tool is constructed using a combination of Java and C++/OpenGL to build an application consisting of both an interactive 3D workspace and a 2D GUI menu/dialog system, which in combination provides all of the necessary tools to enable users to either

build interfaces from existing components or to construct new components from basic pieces by creating specific actions and sensitivities for items in their world. This combination of easy to use features creates a solid development environment from which various new interactive interfaces and components can easily be constructed.

# *Chapter 6: The Set of 3D Interface Components*

One of the main objectives for this research project is not only to create a tool to enable the rapid development of interactive three dimensional user interfaces, but also to develop a set of reusable "3D components", each of which is designed to provide a generic 3D tool for handling a common user interaction task. When put together as a set these components provide a toolkit for the quick implementation of a 3D user interface.

Each of these components is intended to provide the shell of functionality needed for a common interface feature, yet be flexible enough to be easily adapted to the specific task at hand, thus allowing the developer to fill out the details to complete his/her task specific interface.

## 6.1 Designing the Set of Components

The set of components initially needs to provide a core set of key interaction components. However before this set can be implemented it is critical to clarify the overall concepts that make the backbone of how the interactive 3D applications will function. This includes the conceptual manner in which the components will be used and how these components will apply to the user. The following sections describe this projects conceptual layout, including how in general the components are intended to function and fit together. Providing details on each component, how the user relates to these components and how the components fit into the overall system.

### *6.1.1 The Key Design Concepts for the Components*

The design and the concepts on which the designs are based represent the most creative element of this research. This chapter does not attempt to explain this creative process (i.e. how the designs are developed, for more on that see *Appendix C*), instead it describes the basic design principles and the conceptual ideas on which the collection of components is based. The following sections describe each of these broad concepts in terms of their overall objective and how that applies to the interactive systems and components. The concepts include:

- **The Spatial Active World**

  Essentially this concept describes the idea of using the three dimensional space to its full spatial effect and where possible incorporating "active" interface elements into that space. The human visual system is far more sensitive to movement that any other feature of a scene and yet most of the interfaces developed make little use of this. This key design principle is intended to make the world a subtly "active" environment where information interactively presents itself to the user, rather than statically waiting for the user to interact with it.

- **The Independent User Concept**

  This concept presents the user in a 3D environment as a mobile independent entity (whether the world be a local machine or the Internet, the user interacts as a stand alone entity, needing to take everything he/she needs with him/her). This concept enables the user to think of him/herself as the central item in the system (i.e. he/she takes everything he/she needs with him like a turtle or camper/backpacker, so he moves around the system/web as an independent person using only the tools he/she takes with him or the tools in place in the world). The line between local systems and remote ones is becoming blurred hence the user needs a solid base from which to work. No longer can he/she think of a particular computer/file system as a base (because he/she may work with many, spread across the web), instead he/she needs an independent sense of self (a self that carries the needed tools and content with him/her).

- **The Multi-Dimensional Component Concept**

  This refers to the need for "components" in a 3D interface to be capable of functioning effectively when viewed from a range of differing perspectives. For example, in a shared working environment, different users will be viewing the 3D world from totally different views, yet they may be viewing the same selection set. Consequently the presentation of that set needs to be designed to be effective from multiple viewing angles.

- **Move With User Tools & World Content**

  Items are available to the user in one of two types:

  - *Move With User* - these items move with the user throughout the world/environment (eg. the tool belt concept is an example of the storage tool for such move with tools). They are always available to the user no matter where he/she is in the world.
  - *World Content* - these items exist at a fixed location in the 3D world/environment and function in place in that 3D location.

- **The Tool vs. Content Separation**

  One of the key issues in creating an effective 3D interface is to aid users in determining the function of items in the world. That is to help identify if an item is a "tool" or merely a static structural item. Snibbe [Snibbe 92] would use the term "Self Disclosure" to describe components that indicate their functionality through their appearance. Where possible "self disclosing" components are ideal, however this will not be possible in all cases. To alleviate this problem, the concept of tool vs. content separation is proposed.

  Essentially this involves (where possible) grouping tools in known collections or places (eg. the tool-belt or Swiss army knife) where the user will know to look when seeking tools. In addition to these groupings the availability (as part of the move with user tools) of a "WhatisIt?" tool can also aid users. This tool's sole function would be to tell the user what an item of content is and which tools are most relevant. This tool would of course be a "move with user" tool.

In order to provide a comprehensive set of interface components, there is a need to develop a large number of components across a broad range of interactions. This research selects a small, but critical sub set of the comprehensive set, to demonstrate the potential that such reusable sets of components can provide.[19]

To create this base set of 3D components, it is necessary, as a first step, to identify the key interaction tasks that users require specific components to handle and within that set identify the best three dimensional methods to employ to achieve those objectives.

## 6.2 Tasks to be Handled

In everyday use the average user will require numerous interface elements to view, specify and control a range of values and types of information. Although there appear to be many interfaces (and interface components) in use they can usually be boiled down to a relatively small set of base level "tasks" that a user undertakes. In many cases these base level tasks are combined in varying combinations to create more complex interactions. For our purposes identifying and providing three dimensional interface systems to handle/solve these base level tasks enables the construction of a set of core components. These can then be recombined, by users, in different combinations to handle a variety of more complex interface and interaction tasks. The particular tasks that this research is interested in include:[20]

- **Selecting from large/medium sets of structured data (eg. hierarchical data)**
  The type of structured data referred to here is that which is categorized into sections and sub sections based on some appropriate scheme. From the interface perspective developers are primarily interested in the effective presentation of such structured data to enable the users to locate or place items in the correct section or sub-section of the overall system. The interface required must also effectively communicate the structure of the system (i.e. to make it simple for the user to understand how it is laid out and how to access the desired target). This type of task is relatively common both in computer systems (eg. locating files in a file system of directories and sub directories) and in other more "real world" tasks (eg. locating items such as books, songs or movies from libraries with categorized sections).
- **Selecting from large/medium sets of unstructured data (eg. search data)**
  An example of the type of data that fits this description would be the results from a web search for a topic. This type of search usually generates a large set of unstructured results (i.e. each result is unrelated to all the other results, and the overall collection is simply a single large set of items). From the interface

[19] As a result of this sub setting process a range of other possible components, ideas and interfaces were identified but not fully implemented.

[20] Note that this is not intended as a comprehensive list of basic user interactions, however it is an attempt to identify some of the most common interaction tasks.

perspective developers are interested in presenting this unstructured data in the form which makes it easiest to locate desired items. With the amount of information now at everyone's fingertips (via the Internet), this type of practical interaction is one that can benefit from the advantage of a 3D spatial interface.

- **Selection from small sets of choices**
  The kind of interaction described here involves presenting the user with a small set (usually less than 10 items) of options. The type of tasks this would commonly involve include selecting a specific tool from a set or selecting an action from a choice of actions (as well as many others). This interaction task is very common (as demonstrated by the widespread use of the 2D GUI menu, which is the current two dimensional component most commonly applied to this task). As a result finding an effective three dimensional interface to handle small set selection is critical.
- **Enabling setting of range values (e.g. setting of floating point values)**
  The simple task of setting a value from within a range. A simple task in some ways and yet the interface required for this simple task is relatively complex in its operation (eg sliders in 2D GUIs). This type of task is one of the key base level interactions and with an effective interface component to set such values developers have a critical piece of the overall interface system.

In addition to these specific tasks this research also deals with smaller scale controls such as simple tools for setting the state of items (eg. on/off switches and simple button type interface tools). Also of interest is an understanding of general 3D interaction principles and from those building tools/aids or even simple design guidelines to help make the interface more effective.

## 6.3 Designing, Building and Testing the Components

Many of the tasks outlined above are handled extremely well by existing components in the mainstream two dimensional GUI of today, and these components are highly refined in achieving their tasks.

Minimum Value Maximum Value

A good example of this is the two dimensional GUIs' slider (see image above), this component is well established for the task of setting ranging type values (somewhere between the minimum and maximum values). However when transferred to three dimensional space the slider is only effective when viewed in an upright manner from square on (i.e. when viewed from a side view a slider is simply a ball with a dot in the centre. The slider is designed only to be viewed from a 2D point of view), thus making it relatively ineffective for general use in a three dimensional space. Unfortunately this same situation applies to many of the two dimensional GUI components we think of as

standards. Sadly, as is the case for the slider, many of the most effective two dimensional components do not transfer particularly well to three dimensional space.

In response to this lack of suitable 3D interface components this research is attempting to take advantage of active three dimensional space to provide new methods for handling these mainstream interactions from within the 3D environment. Along the way perhaps even discovering some methods that are better than their two dimensional equivalents. Consequently the ideas and concepts used by the 3D interface components to handle the above mentioned tasks reflect this overall objective. These concepts incorporate the use of depth (and motion through that depth) as a primary method of presenting a set (either structured or unstructured) of options to the user.

All of the new designs attempt to make the most effective use of depth, however it is most evident in the components dealing with large sets of information. We see this idea demonstrated by the flow component, where items flow out towards the user from the depths of the screen. The depth of the screen enables many items to be visible steadily extending forward from a distance, and although many of these items are indistinguishable at first, as the flow moves forward they become clearer and clearer. This motion through depth focussed interface and all of the other components outlined below attempt to make the most of the three dimensional space and the extra dimension it provides.

### *6.3.1 Building the Components*

Constructing the components involves using the builder tool to create a generic form of each component (see earlier section *Example Application of the 3D Interface Builder* for an example (the Drum component) of how a component is constructed). This generic form then becomes a reusable item or “3D Component” which can be applied and reused in a range of specific applications. The following section describes the range of components developed in this research project.

## 6.4 The 3D Components

The following sections describe each 3D interface component in terms of what they do, how they were designed and constructed and how their usefulness/suitability as an interface is measured. Each component is described in a full descriptive section (contained in this chapter), consisting of a description of the basic workings of the component, its target tasks, implementation and an analysis of its strengths and weaknesses. Each of these components is also analysed through user testing, these results are found in the later chapter ‘*Evaluating 3D SPACE’*.

### 6.4.1 The Flow Component

The Flow Component represents a new interface (or interaction technique) designed for presenting large sets of unstructured data. An example of this kind of data would be the results obtained from a web search for a particular topic. This type of search usually generates a medium to large number of responses all of which have no relationships to each other. The task of locating a specific result from within these large unstructured sets of data can be a time consuming process. The common approach used in modern GUI interfaces is to provide the user with a large two dimensional list of results. This list is usually presented in the form of a scrolling list of textual web links (see left image of Figure 6.1 below). The user is then required to manually scroll through this list to locate the desired item. Depending on where in the set the target is located, this "searching" task may require a significant amount of active interaction from the user.

The flow component looks at this data in a different way. Rather than treating the result set as a static two dimensional item (in the form of a large textual page) that the user needs to move through, the flow component looks at the result data as a flow of three dimensional items (see right image of Figure 6.1 below). This is similar to a river or stream on which the results flow past the user like bubbles on the surface of the water. From an interface design perspective there is enormous potential to present this data in the form of a three dimensional interface and it is this potential, when applied to the search task problem, that the flow component addresses.

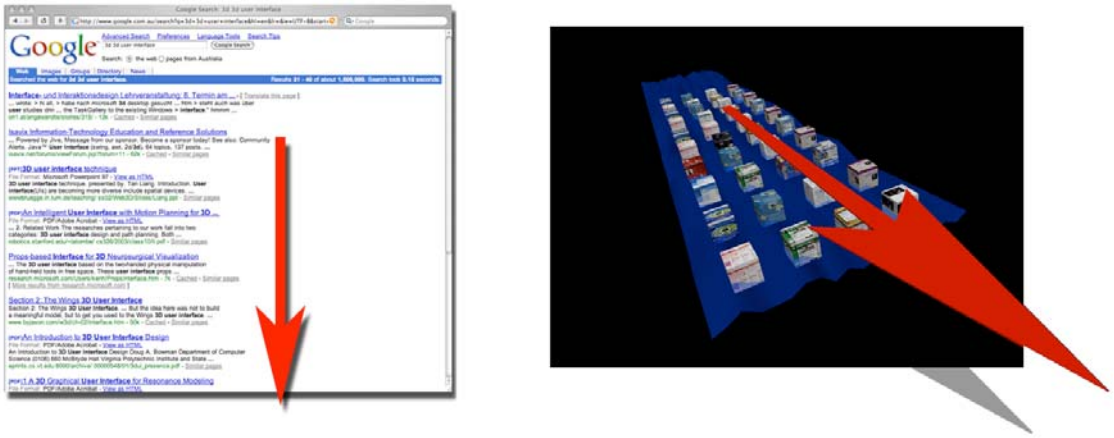



**Figure 6.1: Two Dimensional Scrolling vs. The Three Dimensional Flow**

The original inspiration for this 3D Flow interface came from spending some time observing a three dimensional car/pod racing computer game. In this game (as is the case for many of its type) the screen is used to display a roadway extending ahead of the user, out towards the horizon, and using the depth into the screen (i.e. Z axis) to achieve the perspective effect (see image in Figure 6.2 [Racer 00]).

When a novice player starts playing the game their focus tends to be on the car (in the foreground).Watching it closely and attempting to make the necessary manoeuvres in order to stay on the track and avoid the obstacles. Although this approach achieves

reasonable success it is generally the case that the player is turning too late to make the corners as smoothly as they would have liked.


**Figure 6.2: Pod Racer Game (screenshot)**

As the users became more experienced, not only were they running off the road less, but they also found that they were no longer looking at the car in the foreground, but instead were primarily looking at the horizon in the distance. By focusing their attention further into the distance and making subtle adjustments based on the distant turns/items that they could see coming up, they were able to control the system far more effectively. The thought occurred that this type of flow of items from a distant point through and past the user might make an interesting means of presenting sets of information.

In the game users had found that watching the horizon had been far more effective as it gave them time to look ahead, see the items coming, and adjust to move around them. This use of depth in such an effective way seemed ideally suited to the 3D SPACE project. But would a flow of search results past the user be an effective method for finding a specific result from within a large group of options?

## General Description

The general design of this interface involves taking the set of items and having them flow out at the user from the depths of the screen. The key interactive features, that are unique to three dimensional environments, employed by this system are:

- ***Use of depth*** to display the set of items extending back.
  This allows a larger set of items to be displayed at any one time than would be possible in any 2D arrangement.[21]
- ***Use of movement through 3D space over time*** to enable the set of items to flow past the user
  This is critical in enabling items that are very distant/small (in fact indistinguishable) initially to move steadily towards and past the user. This provides the opportunity to see items of interest in more detail (as they get closer).

[21] Similar use of depth to display larger sets of items is used in [Robertson 91,98], [Leach 97], [see *Visualization Systems* in Background chapter for more]. However those systems do not incorporate the use of movement over time.

The generic form of the Flow component simply takes the set of results and creates an individual item (i.e. 3D representation) for each result. These items are then arranged in long columns each extending back into the screen (using the depth (i.e. Z axis) dimension as seen from users perspective). Putting together this collection of columns gives us the overall result set arranged as a long thin grid of items extending back into 3D space (see Figure 6.3 below).

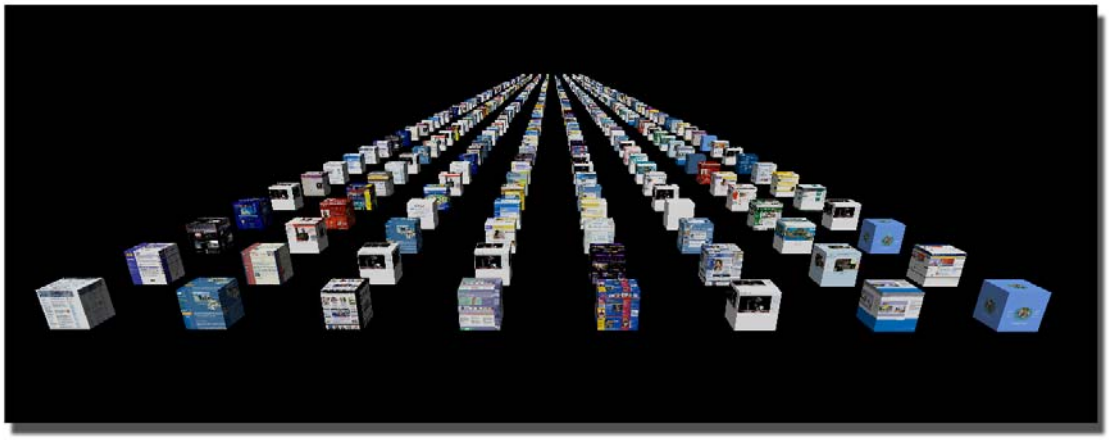


**Figure 6.3: The set of results, displayed in 3D flow form.**

The features that define the shape and layout of this set (i.e. in this case it is a flat plane with eight columns) are flexible to the specific task. For example the columns could be arranged in a circular pattern to create a tunnel like effect (examples of interfaces that make use of this tunnel concept are described in the later sections: Tunnel and Circulatory system components). Although a range of layout options are possible the key feature of any layout is that the items primarily extend back into the depths of the screen. It is this depth that enables the motion of the flow to most effectively use the advantage provided by the three dimensional environment.

The motion of the set is relatively simple and involves the entire set (being treated as a single item) steadily moving forward along the Z axis (depth).

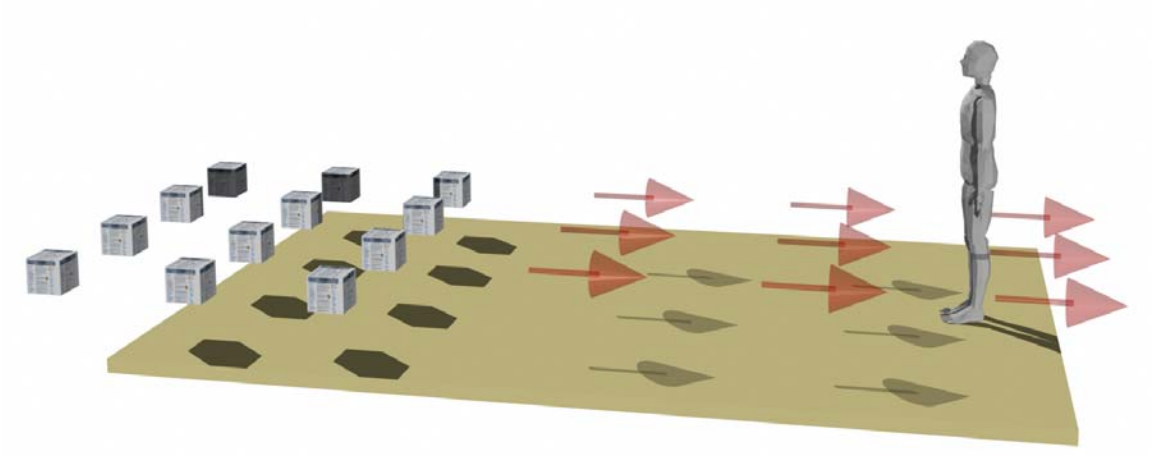


**Figure 6.4: Motion of the set over time relative to the users view**

As the diagram (i.e. Figure 6.4) demonstrates the set of items simply moves through and past the users position (which remains fixed relative to the moving set of items). This simple motion is an effective means of enabling the user to see the items as they move forward from depth, steadily becoming larger and clearer as they do so. With the large set of items able to be displayed the user has a relatively long period of time to see an item coming and hence identify it as the target.

## Presenting the Data

The presentation of the data itself needs to be addressed in two areas, firstly how to present the individual items that make up the set and secondly how to present the set as a whole.

### *Presenting the Individual Items*

The items in the set could be represented by any one of a range of 3D structures. If the data contained results of differing types (eg. images, sound files, text files, web pages, 3D objects etc…) it would be beneficial to present those types as different structures (thus making it easier for the user to identify them). Unfortunately the kind of data involved is generally made up of items of the same type. This makes it more difficult to use the structure itself to distinguish the items.

Perhaps in some cases, such as searches for faces in sets of people, you could use 3D models of the individual peoples faces. Hence each item would be a unique structure and would be easily identified in the set. Unfortunately examples of this type are rare.

Generally speaking the search results are likely to be relatively generic in form and thus require a generic solution. This particular web search problem is an example of one of these generic sets of results (i.e. every result is a web page link).

The obvious thought would be to represent these web links as flat rectangular planes (i.e. 2D rectangles) like sheets of paper with the web page images texture mapped onto them in the 3D environment. Although this is an accurate reflection of what the items represent it does not make the best use of the 3D space. For example when viewed from the side the rectangles will appear to be wafer thin and as such ineffective in terms of communicating any information. Instead of using this flat planar method a better way to present this type of information is to map it onto all six sides of a cube. When presented in this way it does not matter which angle/direction the user is viewing the cube from, he/she will still be able to see at least one of the faces clearly.

**Figure 6.5: The Value of Depth (Cube vs. Rectangle)**

Being able to present the information from a range of angles is an important benefit in that it enables the item (and hence the flow of items) to be viewed from a range of perspectives. This gives the interface greater flexibility (i.e. it can flow below, above or beside the user rather than from a single rigid angle only). Interestingly even in the case where the flow is simple (i.e. flowing from back of screen to front) this cube representation is more effective because it makes use of the changing face that is most visible to the user.

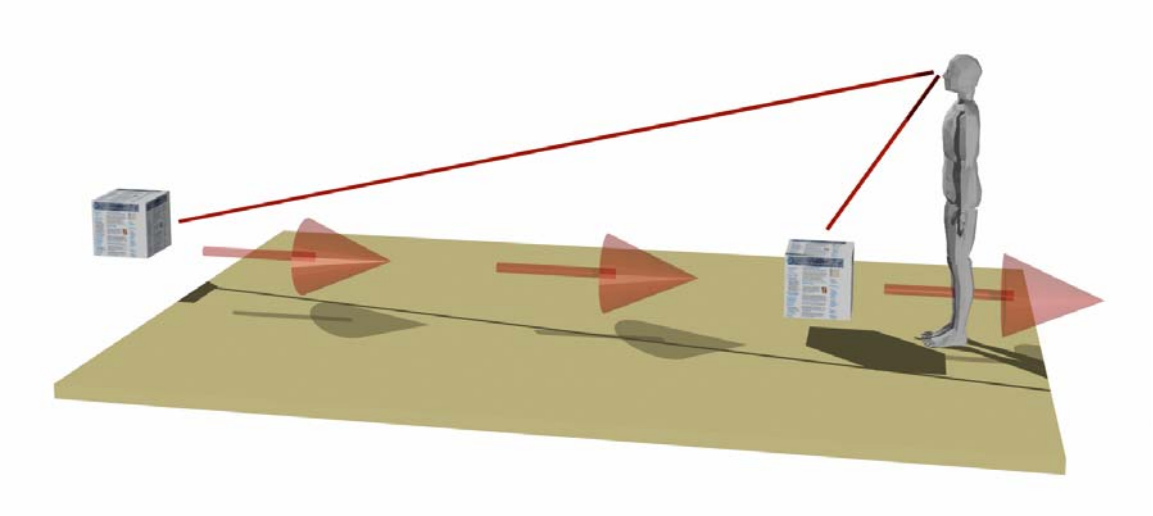


**Figure 6.6: The Visible Faces**

As the diagram above (i.e. Figure 6.6) demonstrates, as the cube moves closer and passes by the user it is more effective in displaying its information (through virtue of its depth, i.e. the user goes from looking at the front face (when item in distance) to looking at the top face when the item is passing under) than the rectangle (which unfortunately gets worse in terms of angle to view as it gets nearer to the user). Hence the cube (due to its depth) is far more effective in displaying its information than its rectangle based equivalent.

The cube/rectangle example is a specific case, however the concept it identifies can be generally applied. That is to say that objects that have shape/volume in all axes (i.e. width, height and depth) are more effective for presenting information in a 3D space, simply because their structure is visible from all angles.

Given all of this information the basic design for the flow component uses simple cube primitives to represent the items in its sets. In the future it is likely that the flow components set of representative items will be enlarged to incorporate other shapes (i.e. to represent types of data other than web page links). However at this stage the simple generic cube is best suited to the task.

### *Presenting the Set*

As described earlier, presentation of the set is a relatively easy task and simply involves laying out long lines of items extending back into space. Although in simple terms this is true there are a number of issues regarding set presentation that are worth reviewing.

One of the first issues and almost certainly the most critical with regards to the effectiveness of the component, is the angle from which the viewer looks at the flow.

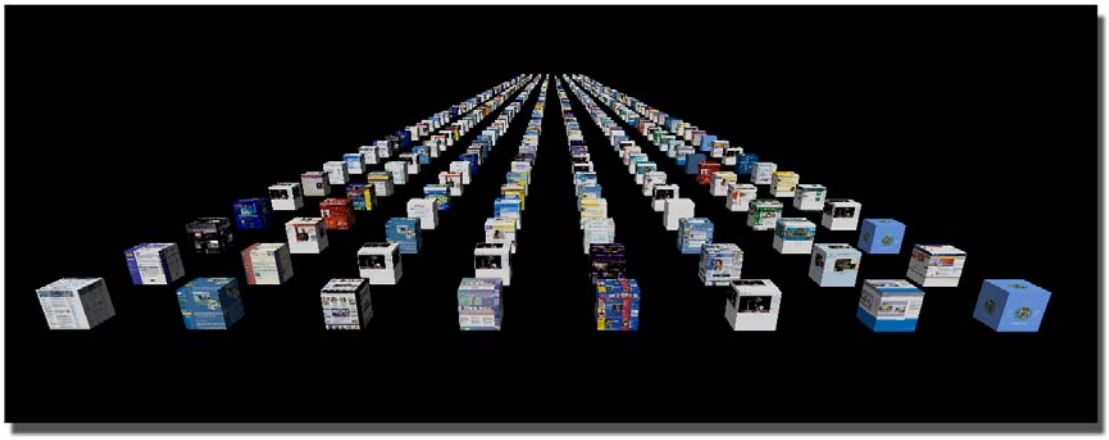


**Figure 6.7: Flow observed from above (flowing through and under user)**

The image above (Figure 6.7) shows a flow viewed from above, so that the items flow past and underneath the user. The fact that the users viewing position (and line of sight is slightly down) is raised from the path of the items generates the visual effect that enables him/her to see the items disappearing into the distance (and towards the distant horizon that is slightly up the screen from his own line of sight). Maintaining this relationship between the viewers line of sight and the sets direction of flow is critical in effectively presenting the set (note that it is not necessary for the flow to go under the user, what is critical is that the direction of flow and the users line of sight go in opposite directions with the line of sight slightly offset to give the slight angle demonstrated in Figure 6.7 above).

Another key issue regards whether the set should be presented as a structure (i.e. sets of columns) made up of its own constituent items, or whether the set should be fitted into a pre existing shell (like a roadway onto which the items are placed). The images on the right (Figure 6.8) demonstrate the basic concepts for the two presentation systems.

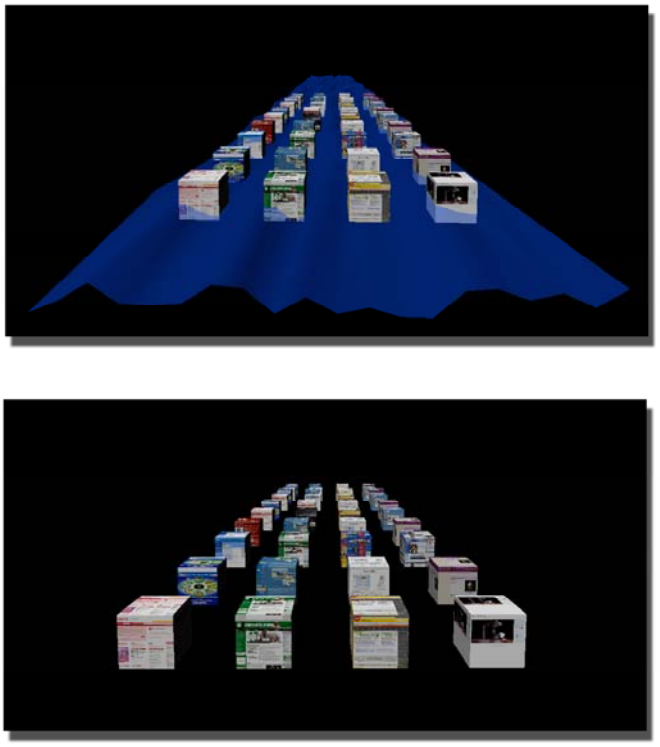


**Figure 6.8: Flow with and without Shells**

Both systems offer advantages, as the images show the shell based version (on the top) gives a better sense of the distant flow (particularly when the number of items is low), however in doing so the structure of the shell itself takes up screen space, makes the items less clearly defined and blocks the view of other elements of the world behind it. Given that the primary objective for this component is the presentation of medium to large sets of data (i.e. sets that contain enough elements to effectively form the shape themselves), the base system assumes that the set is presented without a supporting shell (however adding such a shell is relatively simple and can be incorporated as an optional feature of the component).

For the general application of this component the simple flat arrangement (Figure 6.7 as shown above) where the flow moves through/below the user is used, however this could be varied to flow above, beside or perhaps even have a moving path (i.e. spin around the user). All of these variations are possible, and by simply arranging the set of data as described above this ensures that the component makes the best use of the depth of the 3D environment. With the addition of the motion this enables large sets of data to flow through relatively easily.

## Controlling the Flow

Aside from the basic concept of moving the flow through space towards the user. There is a need for the user to have both the ability and a sense of controlling the flow itself. To achieve this the component needs to provide the user with a means of controlling the motion (eg. in the way that scroll bars and arrows enable users to control scrolling lists). In simple terms the type of controls required by the user include the ability to start, stop, speed up, slow down and perhaps even reverse the motion of the flow. For this purpose and the generic single user system (i.e. displaying results and allowing flow control for a single user) these controls are implemented in a similar way to those commonly seen in video control systems.

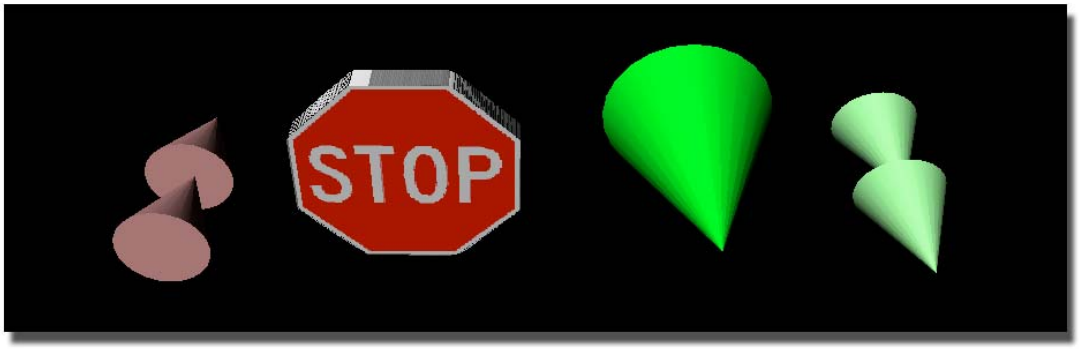



**Figure 6.9: Screenshot of Flow Control System**

As shown in Figure 6.9, these controls simply allow the user to move the mouse (or other pointing device) over the 3D item representing the action to generate the effect (eg. move over the “speed up item” and the flow steadily accelerates). Although this control system is relatively simple and in a sense the control system itself makes little use of its 3D environment. It is effective in its task. Perhaps the application of 3D concepts to control systems of this type represents an area that can be addressed in future research. For now though the focus of this component is on highlighting the use of motion through depth as a means of presenting these large unstructured sets of data.

## The Implementation

The implementation begins with the smallest piece, in this case the generic item. Similar to all of the interfaces in this research this component is designed to provide a reusable generic component rather than a specific one off solution. Consequently the first step was to build the generic search result item. Using the builder application to create this generic item involved making a simple box with an image texture mapped onto its sides and an action so that when the item is clicked it transfers to a URL (i.e. both the texture and URL are arguments of the item and as such can be varied).

With this base item created it is simply a matter of building up larger sets of these (i.e. rows, blocks etc) to create the large result sets as needed for our web search type tasks.

One of the implementation issues that arose during this development stage was that of interactive 3D performance. Although these cubes are geometrically simple the fact that each one is uniquely texture mapped, and the fact that a search result set can contain many thousands of results, leads to a potential system performance issue (i.e. system performance may make the flow redraw slowly and hence not flow past the user as desired). This could be addressed by using clipping planes to eliminate the very distant items and hence reduce the texturing requirements. There are a range of other options to deal with this (eg. only displaying a certain number of rows at any time) however the primary issue is that of ensuring that the flow speed is acceptable. The trials that have been undertaken indicate that most current systems can effectively support greater than 1000 search items with no difficulty (note that the image in Figure 6.7 contains 720 search items and the most distant of those are very small). As a result no active steps were necessary to contain set sizes for this component.

Having built these sets of items the next stage was to build the “parent set/group” item and its links to the flow control mechanisms. The “parent set/group” item is simply a single parent item which includes all of the base items as children, hence when this parent item is moved in space all of its children move with it. With this mechanism all that the control system needs to do is move the parent item along the flow path (see Figure 6.10 for how this is defined) and then stop, start, accelerate etc. based on the users interactions.

Another implementation issue that is critically important to the correct functioning of the component involves determining where in global terms the flow moves from/to in space. Knowing that the flow needs to move towards the user, the simplest step is to take the users line of sight (i.e. the direction he/she is looking) and reverse it, thus giving a flow that would move directly back up the users line of sight from the horizon. Unfortunately this solution does not solve the problem. Because the user is looking squarely down the line of flow (in this system) the items in the foreground will fully block out those behind them and hence the user won’t be able to see the full set to any extent. In order to get the subtle angle that is desired (i.e. where the user is tilted to the flow direction so that it displays items extending back into space and also slightly up the screen, as in Figure 6.7 above), the users line of sight and the direction of the flow need to be slightly offset. In addition to this it is best to offset the flows position a little so that the items do not flow directly through the users personal space but instead flow under, over or to the sides of the user.

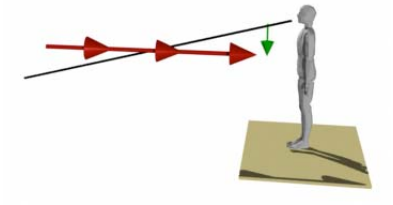


**Figure 6.10a: Get Direction of Flow (offset from line of sight).**

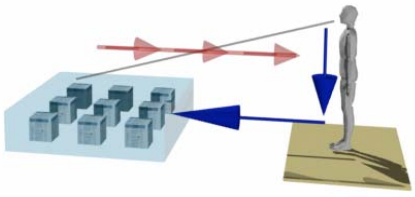


**Figure 6.10b: Position the Set.**

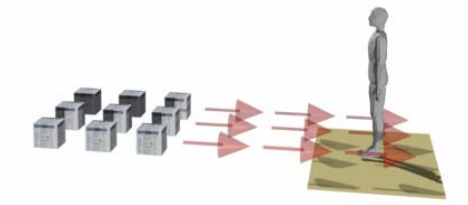


**Figure 6.10c: Final Set with flow direction**

To implement this approach simply take the direction of the users line of sight, reverse it then slightly tilt this new direction down (i.e. relative to the users viewing position with down being negative to the users up vector, see Figure 6.10a). This gives a direction of flow. In addition, use the users viewing position (as a starting point for our set location) and offset it by moving down a little as well as away along the negative direction of flow, this provides a position for the flow to start from (see Figure 6.10b). Using the position and direction for the flow simply place the set (as a single large item) into position and then move it as appropriate along the direction of the flow.

The implementation of this generic component was relatively simple because the generic version requires relatively simple interactions. The complexity of the component is really in developing the specific example where each result item must be individually set with a unique texture map and URL. Although this task would be cumbersome for a human developer it would be simple for an automated tool like a search engine and after all that is the primary objective for the component.

## Summary

The Flow component is a new component created as part of this research project for the purpose of developing a component for presenting large sets of unstructured data to the user. It implements this by creating a long stream of result items that extend forward from the depth of the screen. This stream then flows past the user enabling him/her to view the search results and make the relevant selections. This three dimensional presentation of the data enables the user to view more search results at one time and also provides the user with a new method of searching through those results. For the results of the experimental trials with this interface see the *Results* section later in this document.

## Discussion

One of the observations about the flow component is the fact that it is very specific to the users location and view (i.e. when viewed with the items coming towards you it works well, but if you were viewing it from the other end of the flow (i.e. going away from you) or from a range of other locations it would be of little use). This in turn makes it limited to being an essentially single user viewing experience. The flow component is not designed to be a component that sits in the 3D space waiting to be walked up to and interacted with.[22] Instead this is a component which is constructed on the spot to display data that is generated from a specific user request. As such the component is built on demand for a particular user and his/her orientation. This style of non persisting component (one that is created for a particular task and then goes away) is particularly useful for the type of search result presentation task that the flow is designed to handle. The nature of this search task involves the search engine receiving a request and building a set of results at the time of that request. With that being the case the 3D representation of this set cannot be constructed until these results are available. The flow component can take advantage of this fact by constructing the flow based on the users position and orientation at the time the results become available, thus making the most effective use of the depth of the screen as seen from the users perspective.[23]

Another issue that may arise, when the component is applied to real world tasks, is the fact that the items (representing the web links) are relatively small. Although these items are large enough to see the general appearance of the web page they are usually too small to read the textual content. When searching through sets of similar pages, where the primary variation is the textual content, this may present a problem. This is a challenging issue and one that can be addressed in a number of potential ways, including:

- Using larger link objects. The size of the items is really a variable attribute and hence could be implemented as such (i.e. let the user choose the object size and density). The main drawback, with larger items, is that you can fit fewer of these

[22] The Circulatory system is a component that makes use of similar depth based principles but is designed to remain as an entity in the 3D environment.

[23] This also addresses the issue of texture map orientation, in that the textures can be oriented at the time of creation and hence will be upright from the users perspective.

into a given space and hence there would be less items visible in the flow at any time.

- Enlarging the link objects (in place in the flow) when the user moves the pointer over them. This idea has potential, however when implemented it was found that it was too easy to accidentally move over the wrong item (thus causing it to enlarge) and hence blocking out other items and requiring the user to move away before coming back to try again. This system has potential, perhaps it would be more effective with a less dense packing of items.
- When moved over with the pointer. Move the object across to be above and to the side of the set. At the same time enlarging it to show more detail. This enables the items to be displayed in a clearer location at a larger size. The drawback with this is that it is claiming additional screen space as well as the fact that it still suffers from many of the same selection issues described for the previous point.
- Play an audio track of the textual information when the user moves the pointer over the item. This has the most potential because it does not require any visual additions to the system and it makes an interesting and very effective use of sound to improve the interaction.

Some other thoughts on the areas in which this component could be improved include the thought of using a U shaped flow to enable items to come in and then move out again along the U shaped path, thus providing the user with a second opportunity at locating the item (i.e. if they miss it in the first pass). Note that this may also make the component more effective for use as a persistent type of system (i.e. permanently in the world rather than generated for a particular view). Another thought, with regard to improving the screen time that items in the set receive, is to use the slow in/out method used in animation to give the user more chance to see items that are near the front (i.e. as they get closer they slow a little). This may make it easier to locate the desired items by giving them more screen time while they are in their most advantageous (i.e. nearest) state.

### *The Main Benefits & Risks*

The key issue that the flow component addresses is the use of three dimensional space as a method of presenting large sets of data as compared to the existing two dimensional systems. By using three dimensions it enables the component to display a larger set of items. Due to the perspective effect these items are visible with a varying level of screen presence (i.e. ranging from the near objects (that are clearly visible and have a large screen presence) to the distant objects that are barely present on the screen).

It is possible to display similar large sets in a 2D interface, however this requires representing all of the items with very small on screen representations and this defeats the purpose of presenting them effectively to the user. The fact that the items in the 3D presentation stretch back into the distance (getting smaller as they get further away) is also what enables this set to display the “near” items in larger and clearer sizes. It is clear that this method for presenting a large set makes very effective use of three dimensional space and is more suited to this problem than its two dimensional equivalent. However as a rigid group of items, this large 3D set is not all that useful, given that the distant items are too small to effectively work with. Fortunately this problem can be overcome by

introducing motion into the set and having it “flow” past the users location. Thus over time every item in the set moves from its distant location into the nearer and more easily viewed locations.

Interestingly this flowing system, which is one of the flow components strongest features, is also one of it biggest potential weaknesses. The key advantage of this flowing system is the passive nature of the viewing activity that the user undertakes (i.e. the information comes to the user, rather than him/her actively scrolling through it). This is good because the user is not required to intervene unnecessarily to achieve his/her task, but at the same time it runs the risk that the user may lose a sense of control (i.e. the flow is doing its own thing and he/she has no control over it). Maintaining this sense of user control is a critical feature of any system and hence one of the keys to making the flow component effective is to find that critical balance between the active presentation of the flow and ensuring that the user has a sense of controlling the flow.

Fortunately the concept of a flowing system is not entirely new, in fact most users will have some experience with flow type systems in the form of music/film/video. All of these types of commonly used information are inherently structured in the form of a flow. Thus presenting the concept of manipulating and controlling the flow (i.e. to stop, start, speed up etc) will not be to alien to users. In fact the control mechanisms used by this component will be widely recognized through their use in video and music interfaces. Although there exists this risk of losing control, and its association with using the flow feature of this component, the overall gains to be made through the use of the depth element of the 3D environment outweigh those risks.

As a final point for this discussion section it might be beneficial to briefly look at the benefits of a generic reusable component of this type and how this component would function when applied in the operation of a real search engine. For example imagine that a user comes to your web based 3D search engine, he/she then enters (or speaks) some search text. The search engine then carries out the search of its data base for results. From this point the search engine begins creating the new instance of a “results flow component”. To implement this using the 3D Flow component the following steps would then be undertaken:

```
// Create the shell component

Run the 3D Interface Builder Application
Load a generic 3D Flow component

// Setup the parent Item

Set the components “User location” and “line of sight” attributes
Component uses above to determine location and flow direction[24]

// Create full result set

For (each search result)
```

[24] See Implementation section for how this is achieved

```
{
    // Create this result item

    Load a generic result item
    Set Items URL link
    Set Items Texture
    Add item to Parent Set
}

Output this 3D Flow Component from 3D Interface Builder Application
Insert this instance of the component into 3D world for user interaction.
```

As these steps demonstrate, the process involved in making use of the generic flow component to generate a specific instance of a search result set is relatively simple[25]. This type of automated system is ideally suited to application in real world search engines. The combination of the builder tool and the generic flow component provides the base from which search engines (or a range of other systems) could easily implement a functional 3D result presentation system.

[25] In fact it would be possible to pass the arguments (as described above) into a version of the builder tool (this would be a command line based version which would not need the interactive user interface but would instead simply build components based on command line arguments) and have it automatically build the desired flow component and save it.

### *6.4.2 The Tunnel Component*

The tunnel component represents a simple extension of the flow component described earlier. Functionally these two systems work on the same flow of items from a distance principle. The essential difference is that the tunnel presents the flow of data not as a flat plane (as in the flow component) but as a tube of items extending back into the screen (as shown in Figure 6.11 below).

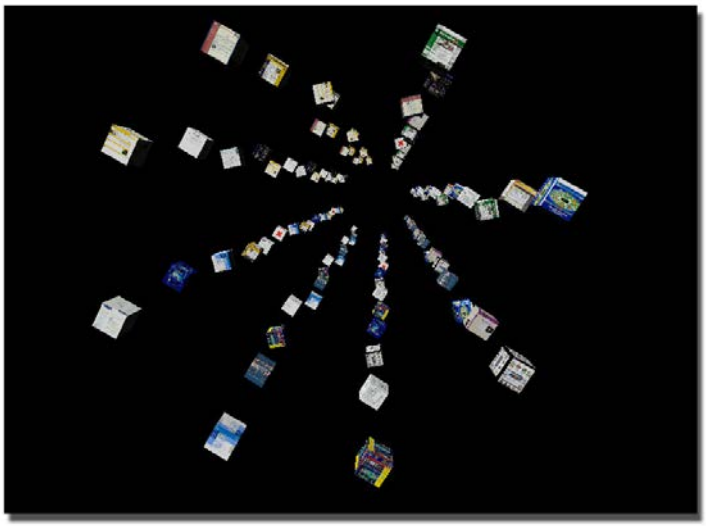


**Figure 6.11: The tunnel component.**

## General Description

The concept of using a three dimensional tube of information is not new with systems such as [Wakita 03]'s INFOTUBE and the 3D workspaces described in [Leach 97] and [Robertson 00] describing interfaces consisting of tunnels or tubes of data extending into the screen. However those systems all use static tubes into which items can simply be placed and later retrieved (i.e. more of a storage space). The innovation introduced by this component is that it implements an active system/tube of data that flows from the depths of the screen through and past the users position on all sides. Presenting the information as a flow, the items on the distant horizon which are initially difficult to work with steadily move forward into visible and functional range. Given the success of the flow component in presenting sets of unstructured data (in the form of search results), this tunnel component, which utilizes the same interactive techniques, provides a similarly effective system and is also capable of presenting larger sets of data than the earlier flow component version.

The tunnel is made up of a series of rings of items, each item representing an individual result in the overall unstructured set of data. Represented as simple boxes these items are arranged so that each ring of items is standing upright to present the user with a circular face of items. These rings extend back into the depths of the screen creating the appearance of a tube of data (see Figure 6.11 above). Similar to the flow component, these rings flow forward towards and past the users location. Given that the tube is most effective when flowing towards and under/around the user, the tunnel component (like its

relative the flow), is created based on the users viewpoint and thus is most effective for presentation of immediate and non-permanent data (eg. search results are an example). To achieve this a similar calculation of flow (based on the users line of sight and a subtle down angle) is used to determine the optimum flow direction for the set.

One of the issues that the tunnel faces, which was not present for the flow system, is that of ring size or radius. The tunnel is a fully enclosed space and as such the users view needs to be contained within that space; not only the users view but also the users sense of their personal space (i.e. if items in the ring are flowing through an area the user would consider to be where his/her body would be, this causes and undesirable lack of realism in the users mind). Given that the tube extends away from the users viewing location the size of the ring of items is important in that if the ring is too tight the user will find that the items appear to pass through his/her personal space, yet if the ring is too loose the items may expand outside the users view too quickly.

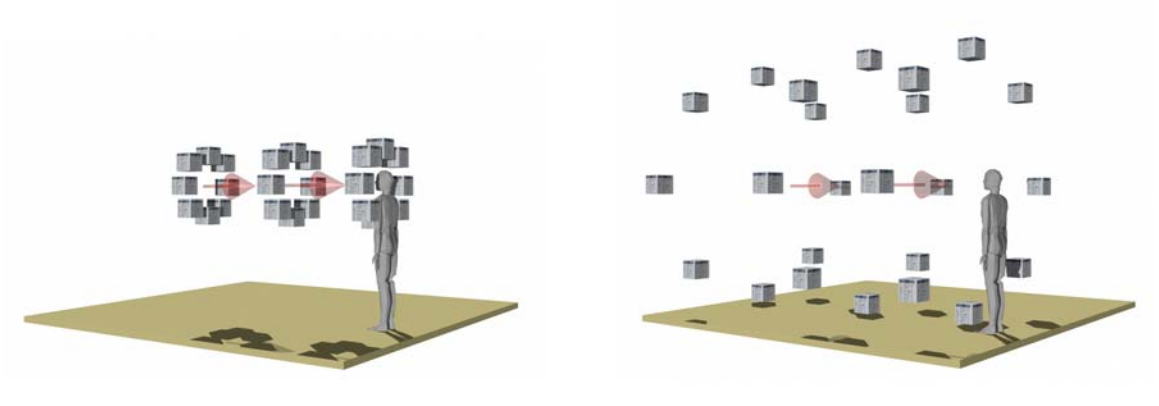


**Figure 6.12: Relative ring sizes.**

As this component is being created for a particular viewing location, and usually an individual users view, this makes it possible to construct the rings so that they surround what could be considered to be the average users physical space (or sense thereof) without closing in too close or extending away too far. For this system to be applied in cases with multiple users presents a more difficult challenge. Given that the flow itself and its direction are based on a single users view, it is difficult to see how this could be applied for multiple viewers simultaneously (without moving each viewer to a location inside the tunnel).

The tunnel enables the presentation of larger sets of data (through the use of a ring rather than a flat plane) using the same depth oriented flow as described for the flow component. This tunnel component presents more data but also uses more of the users view to achieve this, making it an ideal interface for a system where the users entire attention is on this selection task (i.e. as the tunnel consumes the bulk of the user view). For example for search results with large sets of data a tunnel could cycle through considerably more items per step than a flow and yet both provide similar levels of effectiveness in terms of maximizing the use of 3D space.

## Summary

The tunnel component is a simple extension of the flow component system. Constructed in a reusable form this component can be applied to large set selection tasks along similar lines to its relative, the basic flow component. Making use of the benefits offered by the depth of the screen, the tunnel offers developers a variant on the flat planar flow. The tunnel component takes up a larger portion (and is more enclosed) of the users view, however in doing so can present more information more quickly. Clearly these “flow systems”, although functionally designed for the same task, can be effectively applied in different real world scenarios (i.e. the tunnel is more useful when the user wants a presentation of data that takes up his/her full attention/view, where the flow can be used in a less focussed manner).

The tunnel works by extending the use of the concept of a flowing system coming from the depths of the screen by extending the planar flow component into a tube of data. This enclosed tunnel is able to present the user with a tube of options that surround and flow past his/her location. This use of not only the depth but also the width and height makes this component more capable of presenting larger sets of data more quickly.

## Discussion

The differences between the tunnel and the basic flow are small and it is questionable whether the ability to present more items improves the functionality of the system. For example will users simply slow the flow down to view more items or can they simply be presented with a higher density planar flow component. The key benefit that the tunnel has is that it is using a greater region of the viewing area. In the scenario where a developer could utilize either a “more dense flat flow” or a “tunnel”, the tunnel is more effective because the items will be larger than is possible for the planar flow (as the tunnel has more space in which to fit them). This in turn means that more items will be visible further back into the distance and hence the tunnel is capable of more effectively presenting such sets.

On the whole the tunnel makes more effective use of the viewers available viewing range and as such is capable of presenting larger sets of data more quickly. For small sets it may be less effective as the number of items needed to form the shape will not be available and hence the introduction of a shell (eg. tube to outline the shape) may be beneficial.

### *6.4.3 The Orientation Aid Component*

The Orientation Aid is essentially an attempt to provide the user with the most effective means of indicating his/her orientation in the 3D space. The immediate thought that comes to mind is why would this be of benefit, surely your view of the world tells you your orientation. Unfortunately it turns out that this is not the case. The disorientation often associated with 3D interfaces (especially immersive VR systems) is not fully understood. Experiences related to this problem can occur when a user is exposed to a virtual world in which they have full six degree of freedom movement. Initially this system provides no problem to the user as he/she is oriented in an upright manner, however after only a brief period the user finds that they have become disoriented. This disorientation is not related to the users understanding of their state in the "real world" but instead relates to the fact that the "virtual world" and the visual (and other) cues it supplies do not match the users other senses (eg. gravity in the real world is saying that a different direction is down). The user no longer has a feel for where the "down" direction is in the 3D environment, and although they can move in any direction, they have no sense of which direction is which. This disorienting experience is extremely common, in fact there are numerous studies into it and how it could be avoided including [Viire 97].

After moving away from the machine and recovering their sense of orientation users generally question whether this disorienting experience could be avoided or minimized in any way. Perhaps a "visual aid" would have helped them to gain their orientation. Knowing that visual orientation aids of this kind are widely used in the aviation industry (eg. the artificial horizon has been a standard feature on aircraft for many years), both in real physical aircraft and the computer simulations of those aircraft. In all of these systems the artificial horizon is a critical tool in aiding pilots to avoid disorientation and the accidents that it can lead to. As a general rule you would think that an artificial representation of the ground would be of little use, after all you can look out the window to see where the ground is. However there are many cases where the true ground is not visible (eg. at night when both sky and ground are dark). It is in these cases that instruments such as the artificial horizon are beneficial.

It is based on this experience (and the obvious need for better sense of orientation in 3D computer environments) that the range of components described in this section have been developed. All of these components are tools intended to provide that critical sense of orientation in space. Many versions of artificial horizons have been implemented in VR systems prior to this, leading back to the earliest form of computerized flight simulators. However this research aims to provide a generic re-usable component capable of displaying the orientation information in a range of differing forms that can be selected for use in particular instances. Essentially the concept of an orientation aid is not new, however the need for any 3D interface library/toolkit to have such a component is obvious. As such the concept and implementation of such an aid as part of a larger interface toolkit (i.e. as a re-usable component), with a flexible form of presentation is introduced here.

Given that the artificial horizon (as used in aircraft), is a widely used real world system for presenting this exact type of information, it was an ideal starting point and would

represent the base from which the range of new orientation aids, described below, were developed.

## General Description

The general design of this interface involves taking the orientation of the user, relative to the world in which he/she is located, and presenting this information in an easily understandable visual form. The issue of disorientation is something that is specific to 3D systems and hence the solutions are also very specific to 3D environments. The real world example system used in this research is the aircraft artificial horizon (see Figure 6.13). Essentially this tool uses a simple ball, with the base half coloured brown (indicating ground) and the top half coloured blue (indicating sky). As the users orientation changes relative to the ground plane the ball rotates in place. Hence if disoriented the user can quickly view the artificial horizon to get a sense of where the ground is relative to his view. Looking at this system it is clear that its key purpose is very simple:

**Figure 6.13: A "Real" Artificial Horizon**

- To identify where the ground is relative to the current view.

Although this device could present a more complex set of information (including directional information eg. compass headings etc), it does not. This demonstrates the point that more information is not always more useful. In fact in this case the simpler the method of presenting the key information (i.e. where the ground is) the better.

The artificial horizon described above essentially presents a simple two dimensional image showing the ground and sky hemispheres. Although primarily two dimensional this system is representing a 3D concept (i.e. orientation in 3D space). The challenge for the orientation aid component is to develop similar (completely 3D) methods to effectively convey this type of information.

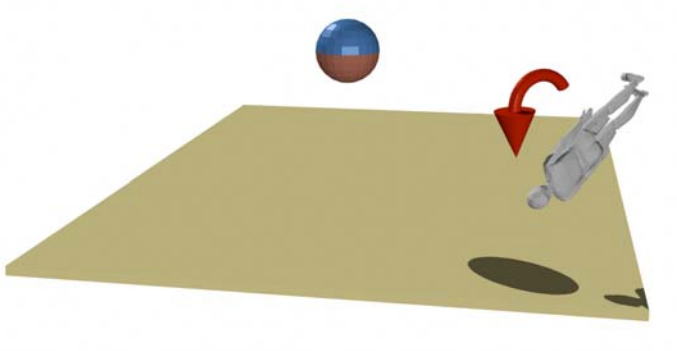


**Figure 6.14: Actual state of world**

The diagrams (Figure 6.14, a, b) demonstrate how these orientation aids work. Essentially they are small devices which move through the world with the user. As the user reorients himself relative to the world they reflect those changes. As the example shows the users view of the world itself can be confusing (i.e. when no objects are visible it is difficult to determine orientation from the view itself, see Figure 6.14a). It is in these cases where the orientation aid is beneficial (eg. in Figure 6.14b the view with the artificial horizon gives an effective sense of orientation despite the world itself being empty).

In the case of the artificial horizon the specification of ground and sky is simple and this effectively represents the state of the real world (i.e. the real world has a ground). Unfortunately 3D interfaces can exist in worlds where the concept (or the physical presence) of a ground plane does not exist. In this type of environment the user still needs to be able to obtain meaningful orientation information. With this in mind the objective is to identify the type of situation where this aid will be used, what information the user is seeking at that time and how that information could be presented. Some of the information that a user may find valuable when he/she looks to an orientation aid includes:

**Figure 6.14a: Users view without aid.**

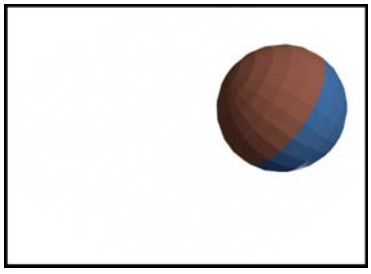
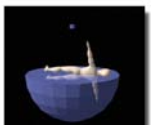


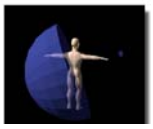
**Figure 6.14b: Users view with aid.**

- A sense of where his/her presence is in the world (i.e. where am I?)
- A sense of where down is (i.e. gravitational direction), relative to current view (i.e. where is the ground?)
- A sense of direction (i.e. where is north, south etc?)
- A sense of where a known state is (eg. how do I get back to upright?)

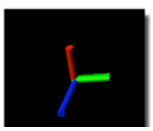
Essentially these points require a method of presenting the users presence relative to the world. This can be visibly achieved a number of ways including (the diagrams on the left below demonstrate how these devices would work):



Display a fixed world with a moving user representation. This is the world centric view (i.e. the world is always upright and the user is moving relative to it). This is easy to understand as people naturally think of a static world in which they are moving.



Display a fixed user representation within a moving world. This is the user centric view (i.e. the user is always upright and the world is moving relative to him/her). This most accurately reflects the true state of the environment relative to the user (i.e. from the point of view the user is looking at, his body is upright and the world is on an angle).



Display a simple changing world representation device (i.e. no user element). This is a simpler system (it is conveying the same information as the above point only leaving out the fixed user representation) and most closely matches that of the artificial horizon.

Display a true relative movement system (i.e. a user representation which moves relative to a world which also moves). This type of system is extremely complex, probably too complex to be of great use. Keeping in mind that simple clear presentation of the key information is the goal, this system (although able to represent more complex relationships) is too difficult to understand to be effective as a mainstream visual aid.

Each of these devices presents the desired data, however the methods are quite different. For an assessment of the effectiveness of each system look at the results from the interactive user trials in the results section later in this thesis.

## The Implementation

The key issue in understanding the implementation of the orientation devices involves understanding the key relationship between world coordinates and user coordinates.

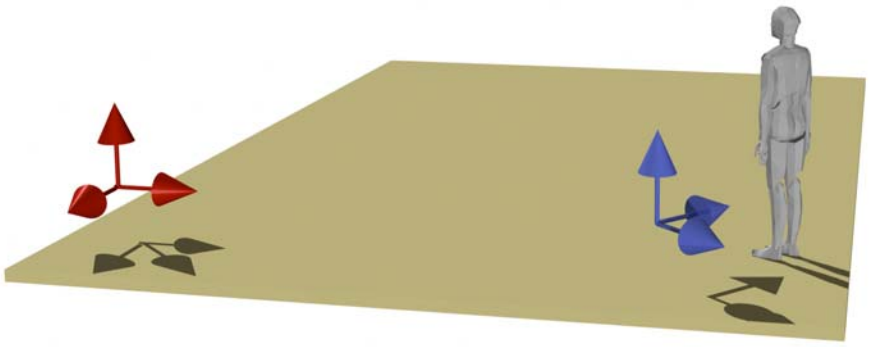


**Figure 6.15: World & User Coordinate Systems**

As the Figure 6.15 shows these are two independent coordinate systems, and in theory each can move or rotate in space independently of the other. In reality however it is usually the case that the world coordinate system remains relatively stable, while it is the users coordinate system that is moved and rotated around within that 3D space.

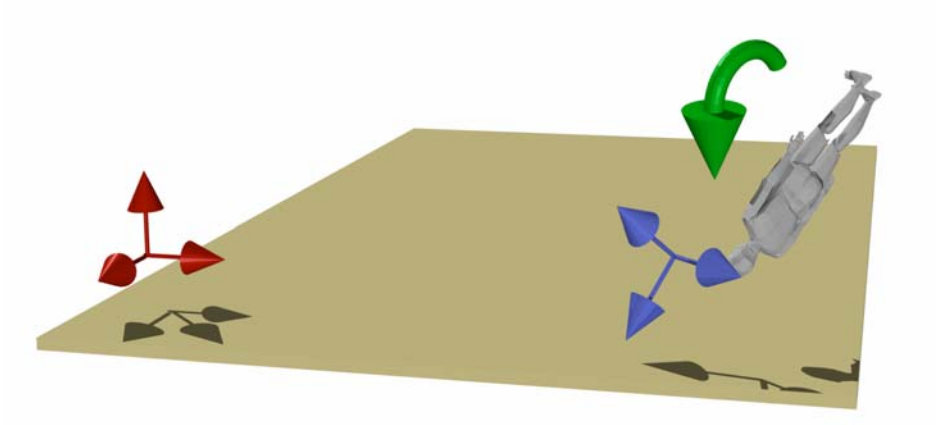


**Figure 6.16: User Coordinates Rotate relative to World**

The interesting thing about this situation is that the perception of orientation is dependant on the coordinate system from which you are viewing the world. That is to say when someone views the world from the external world coordinate system (as in the view of a reader of the previous diagram i.e. Figure 6.16) it appears that the user is rotating and the world is static. However from the users viewpoint (i.e. users coordinate system) it appears that the world is rotating around the user. Hence the sense of orientation is relative to the perspective from which you view the world.

Consequently the orientation aid needs to take this into account. In essence this is relatively simple and involves presenting a device displaying the system that you are not viewing from (i.e. if viewing from users perspective, device needs to display the worlds coordinate system and vice versa). From the viewers perspective (as this is the one of primary interest) they are interested in knowing what the worlds state is relative to their state.

As described in the section above there are a range of methods for implementing such an orientation aid device. This section looks at how these devices function and how they can be created.

Knowing that (from the users perspective) it is the world orientation that we wish to display. It is possible to convey all of the desired information by simply presenting a single device showing the user the state of the worlds coordinate system. Essentially this is exactly what the real world artificial horizon does. To achieve this the orientation aids can represent the state of the worlds orientation using a range of possible physical shapes/items (essentially any item is possible but one which clearly identifies the differing axes is likely to be most relevant), some of the most effective include:

**Figure 6.17: Some examples of orientation representing devices**

Implementing these items as functional interface devices is a relatively simple process. Firstly it involves constructing the shape of the item itself and ensuring that the shape is oriented in the desired manner (eg. if its an artificial horizon ensure that the ground hemisphere is down). The shape is then linked to the worlds coordinate system (i.e. item is placed in the world with its axis matching those of the world system). From this point the device is attached to the users location in space (i.e. so that it moves with him/her through space). This link is achieved by placing the item along (and slightly below) the

users line of sight. This ensures that no matter where the user moves to the item remains visible.

Although the device is linked to the location of the user, it is not linked to the users orientation. Hence when the user spins in space, the world appears (from his/her perspective) to rotate around him/her. As the device is mapped to the worlds coordinates it also appears to rotate (i.e. matching the rotation of the world). From the external perspective it is clear that actually the user is rotating and the device/world that is static. However from the users perspective he/she sees a world (and device) rotating relative to his/her point of view.

Some of the devices described in the above sections incorporate elements to represent both the worlds orientation (as described in last paragraph) and also the users orientation (i.e. these are more complex systems showing both the world and users states at one time).

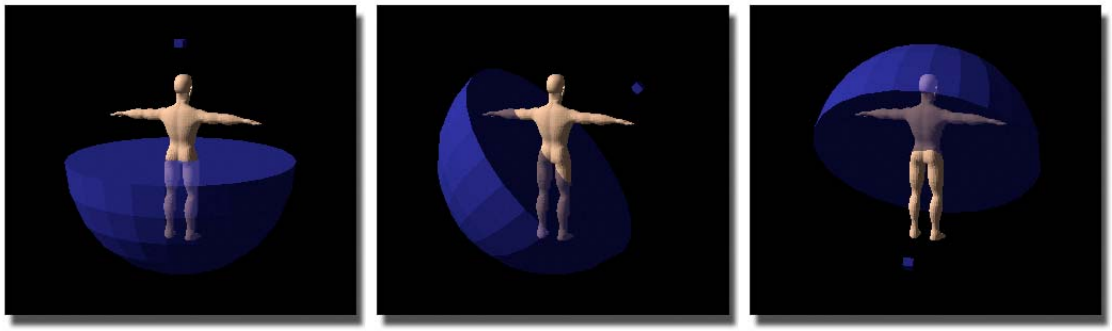


**Figure 6.18: Dual System Device (showing both world and user coordinates)**

This is simple to implement and involves simply adding another element to the basic device to represent the users orientation (in the diagram above this is the figure of the man). This element moves independently of the other world coordinate element (the blue hemisphere) and with this the two elements (in combination) give a sense of one systems relative orientation to the other. The reality is that the device, indicating the users coordinate system, is fixed (notice how the figure of the man remains oriented in a fixed upright manner) and will not move (the users orientation from his/her perspective remains constant i.e. his right hand is always on his right side irrelevant of how he is oriented in space).

This two part system is interesting because it is a more complex way of presenting the same information. Although the complexity may benefit, by providing some extra contextual information, it may also detract (i.e. making the system more confusing, and less effective).

Aside from the device itself, one of the key implementation issues revolves around how the device is presented to the user. For example should it always be present in the users view or should it be something that comes up when requested. The other question is where to put it in relation to the users view (i.e. in the centre of his/her view or off to the side). From an implementation perspective the objective is to make the item noticeable

(i.e. the user needs to be able to see it and make use of it), but at the same time the user does not want it to clog up his/her view (i.e. blocking out large parts of the world). For a more thorough discussion of these issues see the Discussion section below.

For the purposes of this particular implementation of the system, the device was implemented as a small item that is permanently present in the base/centre of the users view. The attributes of this device (i.e. type, size, how often visible) can of course be controlled and it would be simple to build a system that enabled the user to set his/her own preferences with regard to placement and visibility.

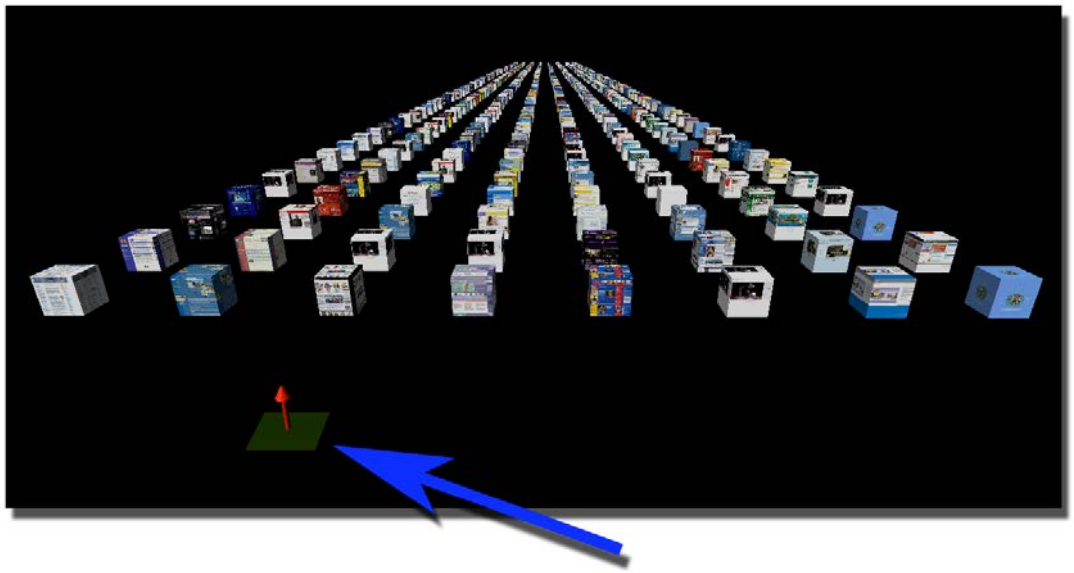


**Figure 6.19: Example Orientation Aid in use.**

## Summary

The orientation aid is a simple, yet powerful tool which forms part of the set of “move with user” components. These tools are not fixed to a location in 3D space, instead they are tools (or items) that a user carries with him/her throughout the world/worlds. In the case of the orientation aid it is a small item, somewhat like a personal pocket compass that the user can carry with him/her and view as desired. The orientation aid does more than your average compass. In addition to the flat directional features of a compass it also provides the user with a sense of orientation in all three dimensions (i.e. incorporating up/down). To achieve this it uses the display of a simple three dimensional device which works as a visual assistant providing the user with information on the state of his orientation with relation to the world.

From the developers perspective this generic “orientation aid” provides a simple base component which can be easily added to an interactive system to provide the user with a sense of his/her orientation. By simply linking the component to the relevant coordinate system (i.e. world coordinates for the common user view case), the developer can very quickly add this feature to an interactive system.

The primary reason that this tool is implemented is the fact that disorientation is a known, difficult problem in three dimensional spaces. By providing an easy to use generic component, which incorporates a range of variants on the basic artificial horizon concept, this tool aims to provide an effective means of indicating the users orientation.

## Discussion

One of the most interesting issues with this tool and all of the family of "move-with user tools" is the concept of how to most effectively display them to the user. The general concept of the orientation aid is that it is the type of component that you want to have on hand at all times (i.e. constantly up and showing the state of your relative orientation). Unfortunately in doing so users must permanently commit a piece of the visual screen to this device. Now if the user is finding this device to be effective and he/she is involved in an activity where the orientation aid is beneficial then there is no problem. However, what about the more common case, where the user is not particularly interested in his/her orientation. In this case the device is not serving a functional task and in fact may be detrimental to the environment on the whole (i.e. it is blocking the view of the items behind it or simply unnecessarily filling up the space). Certainly the orientation aid (i.e. the device) is providing useful information, but having it "hanging around" in the view distracts from the sense of space. Some of these drawbacks can be addressed, for example by using partial transparency we can overcome the visual "blocking" effect. But this does not address the general problem of how to provide the user with access to his/her "move with" tools without needing to have them constantly visible in the view. Toolspaces as discussed in [Pierce 99], [Robertson 00] provide one option, however this places the orientation aid out of the users primary view and a such may reduce its effectiveness.

A similar general approach involves building a special grouping tool (i.e. to contain all of the "move with user" tools). which could be displayed or hidden as desired. One version of this concept is the "toolbelt", where the user has a set of tools that he/she takes with him/her. This belt (a simple structure to store various tools/components) could be either kept on or off screen and called into view as needed. Another possibility is the "Swiss army knife" concept. Essentially this is a single item (always present in the users view), which contains a range of tools and other items. For a more thorough description of this grouping task, and some of the proposed solutions see the Toolbelt component section later in this thesis.

The concept of an orientation aid is not new, in fact there are existing real world systems, such as the artificial horizon, built for exactly this purpose. One of the interesting features of the generic orientation aid component as built in this research is its ability to use a range of different possible orientation representing mechanisms/devices. The advantage of such a flexible system is that it enables the developer to choose the type of device that is most appropriate. However the negative that comes with that is the lack of consistency (i.e. one system uses a different device to another). Surely it would be best to use a single device for all circumstances (i.e. then the user will immediately recognize the device and understand its function). In the long term identifying this single consistent device is the objective. However in the shorter term, one of the objectives of this component is to discover which device will prove the most effective. Hence the orientation aid provides a range of options to enable the easy development of systems with multiple versions and

thus enable comparisons between them. With the overall goal of identifying the best mechanism, which will then be used as the default device.

***The Main Benefits & Risks***

The key benefit sought by the orientation aid is providing the user with a better sense of his/her state in the world. It attempts to achieve this by displaying a device representing the worlds orientation relative to the user. Although the information thus displayed is useful and of interest, does it solve the disorientation issue it sets out to address?

The key problem here is not so much whether the device effectively displays its information but whether the user is looking for this information at all. The first key question that arises is whether the user makes active use of instruments when working in a 3D environment (i.e. when working will the user look at the device). Previous research demonstrates that "real world" instruments of the type discussed can be very effective (i.e. artificial horizon). Although less clearly demonstrated in 3D computer environments, it is reasonable to assume that they would at least provide some positive input to the issue.

This raises the issue of detachment of information from the user. By using tools (i.e. like the orientation aid) to tell the user his/her state, we are creating a detached sense of self for the user. Knowing that the real world already gives the user a sense of detachment from the 3D environment (i.e. real world gravity does not change to match the 3D environment). Does the use of tools or aids to tell the user about his/her state increase that level of detachment? Certainly a more immediate and direct method of informing the user of his/her orientation is desirable, however achieving this with a graphical system alone is very difficult. This orientation aid describes and compares the performance[26] of a number of simple visual tools/methods for presenting this information.

However it is true that perhaps orientation feedback could be better supplied by hardware (i.e. devices) that could, through interactive feedback, provide the user with the desired information (eg. simulator devices that physically change orientation or perhaps simpler forms of force feedback to give gravitational feel). Such a system would obviously be more effective and certainly preferable to any simple visual aid system, however they are unlikely to become mainstream in use. As a result the aids described in this section do not attempt to provide the ultimate system to represent orientation to the user, instead they are a simple method aimed at providing useful information which may help the user understand his/her orientation in space and also reduce or aid the recovery from disorienting experiences.

[26] See section on interactive trial results in the results section of thesis.

### *6.4.4 The 3D Slider Component*

The 3D slider component is actually a collection of different components each designed to solve the need for a value setting mechanism within a 3D interface. Necessity is the primary cause for the development of these components. Essentially when developing the trial systems in this project there existed a need to enable users to control the size of varying items in the worlds in which they were interacting (eg. ability to resize the tool-belt to take more/less screen space). To achieve this users needed a component like the 2D GUIs slider, however there is no three dimensional equivalent. In fact in most cases a simple 2D slider is used by overlaying it over the 3D environment. The need to develop a value setting component that could be inserted into 3D space and function when viewed from multiple angles was clear.

The Line-Slider (see later section in the *Simpler Components* section for a full description) component is designed to take the existing principles described in the 2D GUI slider and make it 3D in structure.

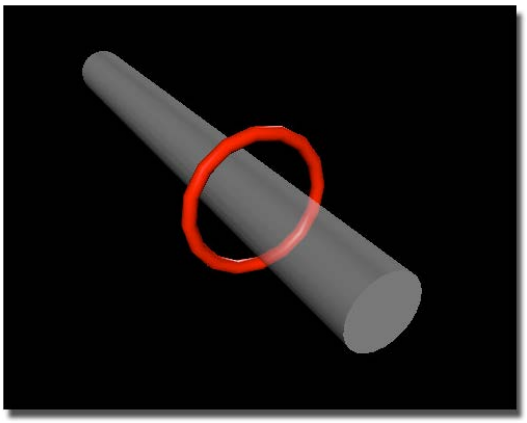


**Figure 6.20: The Line Slider**

Implemented as a simple cylindrical line with a ring slider that moves along it, this component is very effective when viewed from square on but fails badly when viewed from other angles. The failure of this component identified the key need to create a new value setting component that can function (both in terms of displaying its value and enabling the setting of that value) when viewed from any angle.

The key problem with the line slider is the linear nature of the range along which the ring moves. Essentially this makes the movement of the ring very two dimensional in nature. To function effectively when viewed in 3D space this path needs to be three dimensional itself. This can be achieved by taking the Line-Sliders' linear path and making it into a spring or helix based path. If arranged so that it expands out from a central point this spiralling path provides the desired 3D path (as shown below) that can be viewed from all angles and still indicate its value.

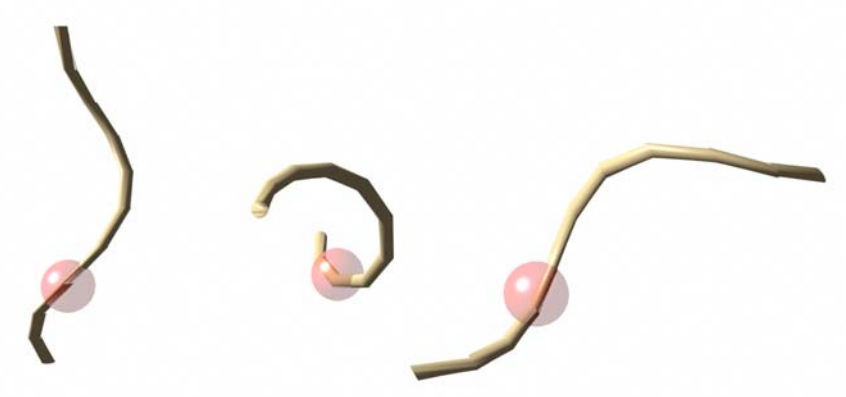


**Figure 6.21: The Spring-Slider Concept.**

Another approach is to step away from the end-to-end line (or path) based approach and treat the task as a volume based system. That is to present the range of values as a volume ranging from zero to a maximum size.

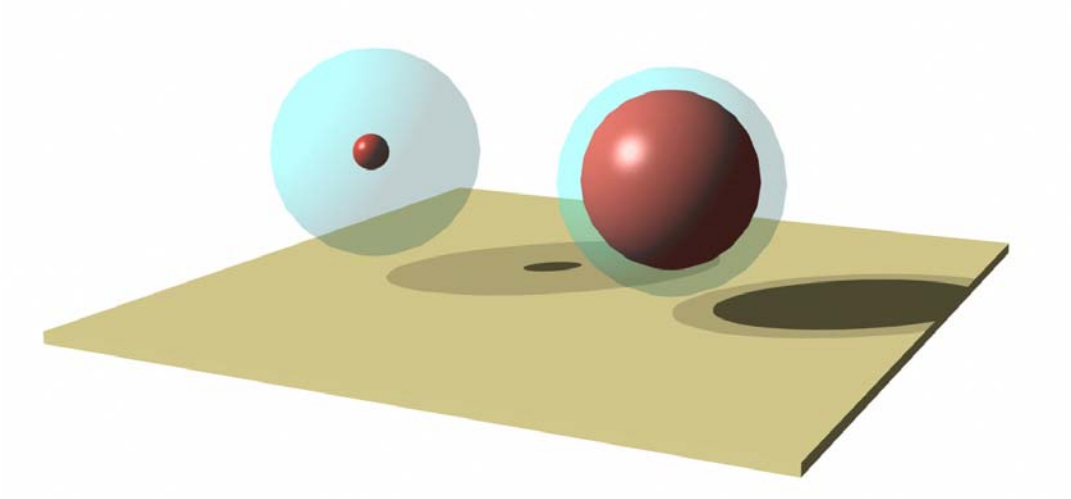


**Figure 6.22: The Volume-Slider Concept**

Such a system uses a “maximum volume” item (eg. the aqua sphere above) which can either be partially transparent or can be a shell inside which a “value volume” (eg. the red sphere above) can be enlarged or reduced to set the value. This system is quite different to existing systems however it is very simple to understand, has an identical appearance from all viewpoints and offers a new perspective on the value setting challenge.

## General Description

Two different types of value setting components are described here. The first type is the volume filling style (as shown above in Figure 6.22), essentially these components set a value by ranging between an empty volume/value and a full volume/value. The second type are the linear value setters which function by setting a point/value along a line between a low (minimum) and high (maximum) value (examples of this type include the Line and Spring Sliders). Both of these systems can effectively set values, however each has its own unique advantages and disadvantages, making them more or less suitable for certain tasks. the following sections describe each component in more detail.

### *The Volume Based Systems*

The volume based systems are relatively simple in terms of how they function. Essentially they detect when the user clicks on and drags the mouse relative to them. Based on the drag direction (i.e. away from centre of volume causes an enlargement and towards the centre causes a reduction of the volume) the volume (or value) is adjusted. This system can be implemented using a “volume object” of almost any type, however the sphere is most suited as it has an identical appearance from all angles and can be enlarged/reduced using a simple scale operation. Other versions could be implemented using objects that are offset from the centre and appear to grow up towards the maximum value. For example a cylinder volume could start small at one end of the “maximum volume” cylinder and grow out and up towards the full size of the “maximum volume”.

This system offers the beneficial feature that when viewed from the side provides a linear style value marker, however it is more complex to implement.

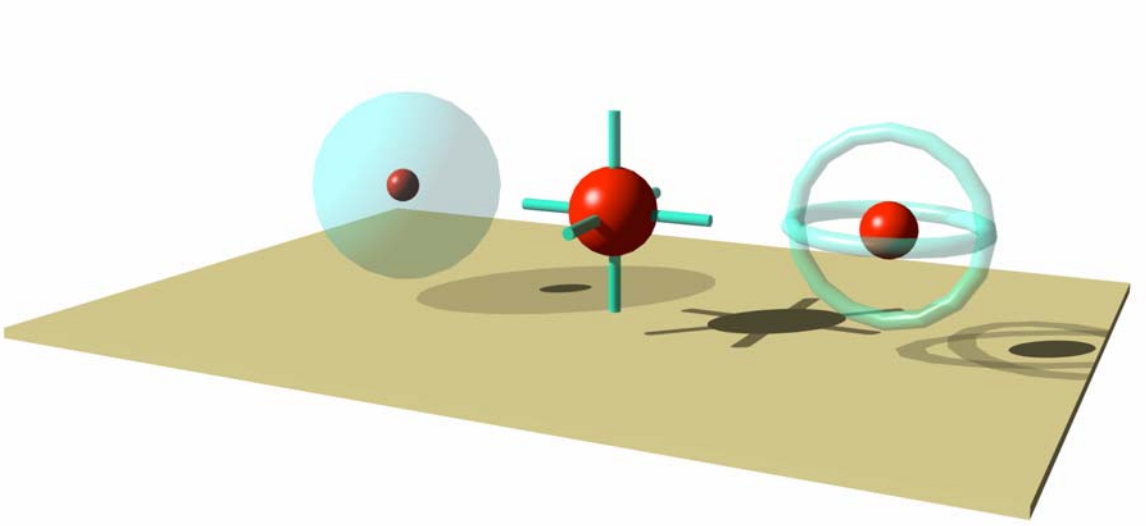


**Figure 6.23: Options for indication "maximum volume".**

One of the key issues in designing a volume based system is to specify what physical structure to use in defining the "maximum volume" size. For example this could simply be a larger version of the value item (as is the case on the left above) which is partially transparent so that it enables viewing of the "value item" inside it. Another option is to use range bars (like an axis device that sticks out of the value item extending to the maximum range as in the centre above (note that these bars can also be used to present value data eg. numbers). The third option is to use a "shell" object that consists of a set of rings which when put together mark the outer range for the value item (as shown on the right above). These systems all function in an identical manner, it is simply a choice for the developer to determine which style is most relevant.

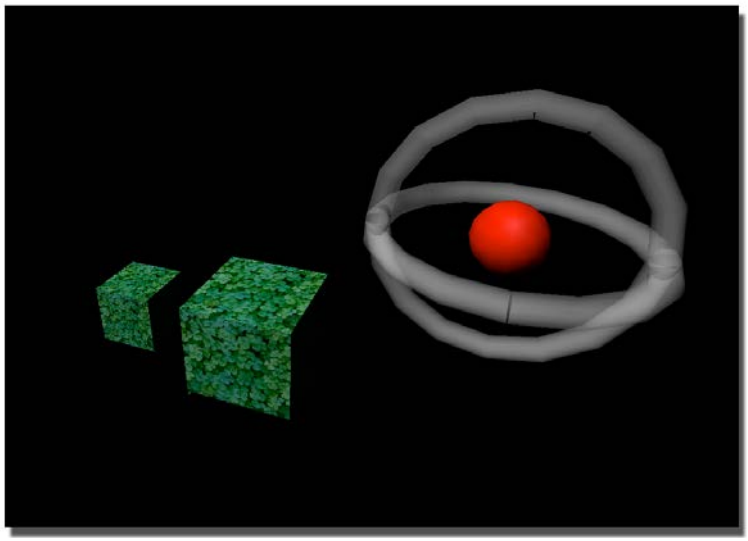


**Figure 6.24: Screenshot of Sphere Slider in sizing task**

Given that all of the volume based systems utilize a size to represent the value, they are particularly suited to setting range values where the result can be thought of as a size or strength (eg. setting the brightness of a light, the size of an item or the intensity of a colour is well suited to these components). However they are less suited to setting specific values (eg. setting a distance).

The fact that these components are equally effective (in fact identical) from all angles is a particular strength and as such makes them the only type of value setting component capable of this feat.

### *The Linear Value Systems*

The linear systems are all based on the basic design as outlined by their two dimensional parent (i.e. the GUI slider). In simple terms they involve the specification of a line between two points (a minimum and maximum value) in three dimensional space. Every point along that line represents a value between the minimum and maximum amounts. In the basic Line-Slider version the line is a simple straight line between the two points, however the line need not be so simple. Curving paths offer the potential to move the path off the 2D plane and make it function more effectively in the 3D space.

The Spring-Slider essentially takes the minimum and maximum points and creates a curling spring based path between them, thus generating a path that can be viewed from all angles in a 3D environment. The actual path used can be defined in a number of ways, ranging from a very simple rotationally growing spring through to other more complex mathematical functions.

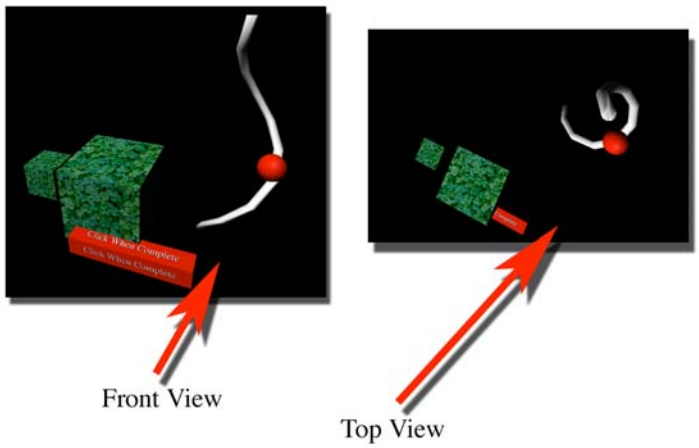



**Figure 6.25: Screenshots of Spring-Slider (multiple viewing angles)**

As the diagram above demonstrates the Spring-Slider can effectively be viewed from multiple angles and still function effectively. These screenshots show the slider as seen from two differing viewpoints (i.e. front on and directly above), these particular viewpoints represent the extremes for this component (i.e. for the Line-Slider the top view is where it cannot work) and as the diagram shows in both views a clear path is visible and it is clear where on that path the ball is located. Unlike the volume based systems the Spring-Slider enables specific values to be easily set and hence is more suited to this general application.

## Summary

The slider is core piece of the functional componentry for any user interface developer. Providing the basic ability to set values within given ranges the slider can be applied in numerous interactive situations. Previous 3D systems simply use 2D sliders overlaid over the users 3D view to enable the setting of such values, and although functional such

systems are limited to single viewer interaction and detract from the immersive sense of the 3D world. This research has designed and created several new components to enable the setting of these values from within the 3D space (see the results section for an analysis of the performance of these components in user testing). The Sphere-Slider (in one of its three forms) provides a volume based expanding system which is particularly effective for setting ranging values where size or strength type values are being set (eg. intensity of a colour). The Spring-Slider provides another component for this value setting task, a variation on the basic linear slider the spring slider takes the straight line used in the line slider and twists it into a spring shape. This spring can then be followed by the value marker to indicate its value. This system is more general than its volume based equivalent and although less effective for some tasks is suited to a broader range of applications.

These components represent a new method for setting these range type values from within a 3D interface. Designed to work from all viewpoints they offer the developer a reusable component which can be applied to many tasks within a 3D user interface.

## Discussion

One of the key challenges for these components is in comparing them against the well established 2D GUI slider. The slider is an accurate mechanism for setting a value within a range, however with these new systems it can be more difficult to place/set the value accurately (eg. when looking at the Spring-Slider it is not as clear exactly where half way is). The GUI slider itself is not perfect in achieving this, however its straight line presentation makes it simpler for users to see exactly what value they are setting. Using a combination of experience with the component, markers along the range and (where possible) an ability to view the effect (eg. see the colour that is being set etc) should solve this problem.

A challenge that is specific to the volume based systems is the fact that it is not possible to “jump” to specific values that are smaller than the current value. This is due to the fact that the “value object” itself is blocking the range and hence to set a value the user must size up/down to get to a value. This is a limitation that is caused by the functional nature of the component and would be difficult to change (although it would be possible in the axis based system to allow users to click on the axis bars to directly set values).

Essentially these components offer a new mechanism for setting values of this type from within 3D space. The methods used are different to those with which users are accustomed, however with experience these systems offer enormous potential. The fact that they are able to be to placed inside the 3D space rather than overlaid enables the user to remain within an immersive environment to set such values. In addition ability to view these components in multi-user (multi-viewpoint) scenario and have all users able to see the values being set opens up the possibilities of collaborative applications.

### 6.4.5 The Drum Selector Component

One of the most common interactive tasks undertaken by users is selecting particular items from a small set of choices. This critical task is handled, very effectively by a number of 2D interface components, the main one being the menu (see Figure 6.26).

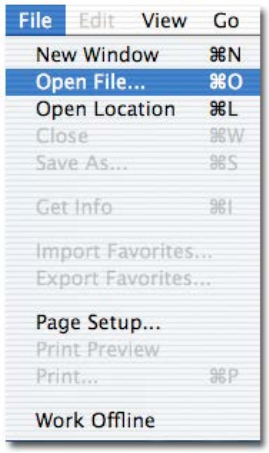



**Figure 6.26: Two dimensional menu.**

The menu can be directly applied in a 3D interface (note this 2D menu in a 3D interface is described in *The Simpler Components - Flat Menu* later in this thesis), however such a system makes little or no use of the potential offered by the 3D environment.

The key concept behind the drum component described in this section is to describe a new 3D component for this task of selecting from small sets.

The need to provide a natural mechanism for viewing the range of options and enabling selection is critical. In 2D GUI systems the menu is the widely used component for this task. Based on the real world menu as presented in restaurants for selecting food. This method is very effective in two dimensions however when transferred to 3D, although still functional, it makes little use of the potential of 3D space. The 3D drum selector proposed here essentially takes the basic principles of this menu style and adjusts it to make use of 3D, in particular rotation of items through depth. A little like a rolling pin on which the items cycle. Inspired by existing 3D interface systems such as the SuperCube [Waterworth 94], which uses a simple box across which differing items can be scrolled, and Holosketch [Deering 95], a system that uses a pie chart like menu overlayed over the users view to enable interaction with it.

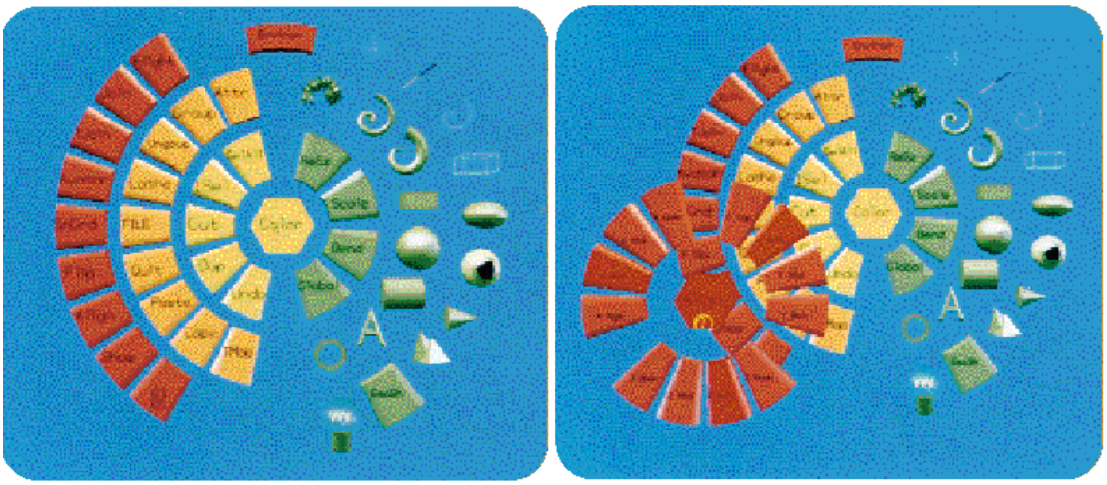


**Figure 6.27: Holosketch interface.**

Both of those systems provide a menu like system for selecting items from small sets within a 3D interface. Despite their strengths, both of these systems have limitations.

Holosketch is essentially two dimensional and is simply overlaid over the users view and the SuperCube although 3D only allows the display of four items at any one time (hence the user must learn that items can be scrolled in from some arbitrary place.

The drum component described by this research takes some of the ideas used in these systems and extends them, incorporating new concepts such as multilevel options through a system based on the “Russian doll”. A Russian doll is essentially a hollow wooden doll inside which is another (slightly smaller) hollow wooden doll and within that there is another and so forth. This “small versions inside a larger version” concept can be applied to sub-levels within a higher level. The drum component also introduces a new 3D method of laying out the choices as well as introducing movement into this selection task to enables users to rotate the set of choices.

## General Description

The drum component takes the small to medium sized set of items and represents them as items within a cylindrical tube of choices. As most menus contain textual information the tube is a long thin shape, however this is flexible to the data being presented. In addition to this basic cylindrical layout the drum also introduces the capability to rotate the set of choices in place, thus enabling all choices to be cycled from the back to the front and vice-versa.

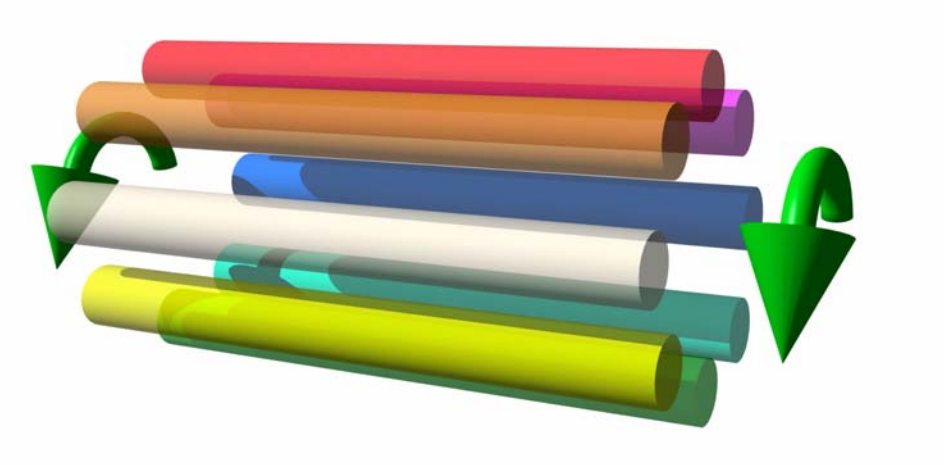


**Figure 6.28: The concept of the drum component.**

One of the strengths of the drum is its capability to function when viewed from any angle. As the following diagram shows when viewed from the front or top the drum appears like a menu which scrolls from the back to the front. When viewed from the side the drum appears much like the pie chart style system outlined in Holosketch. At viewpoints in between the drum presents angular versions. This capability to be viewed and function effectively from all angles is something that none of the preceding systems (i.e. menus, SuperCube, Holosketch) achieved. This capability is particularly useful for

application in multi-user environments, where a single “drum” may be being viewed by multiple users from differing viewpoints[27].

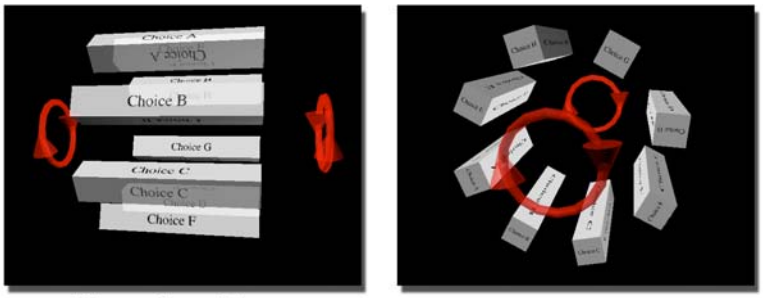



**Figure 6.29: Drum’s effectiveness from all angles.**

Partial transparency is also utilized to enable users to see (particularly when the drum is viewed from a point perpendicular to the front/top) that there are additional options at the back of the cylinder and hence rotating the set will bring them to the front. Although subtle this use of transparency provides the user with an immediate sense that there is more to the set of options than those visible in the foreground (this system is different from the systems such as the SuperCube where the items scroll from a non visible location i.e. for the drum it is clear that these items are simply at the back of the cylinder).

The basic (single level) drum component is fully contained within its rotational space and hence (unlike other systems such as menus), the addition of more items does not claim additional screen/view space. This offers the ability in a comparative sense to use less screen space to display the same number or more items (see diagram below).

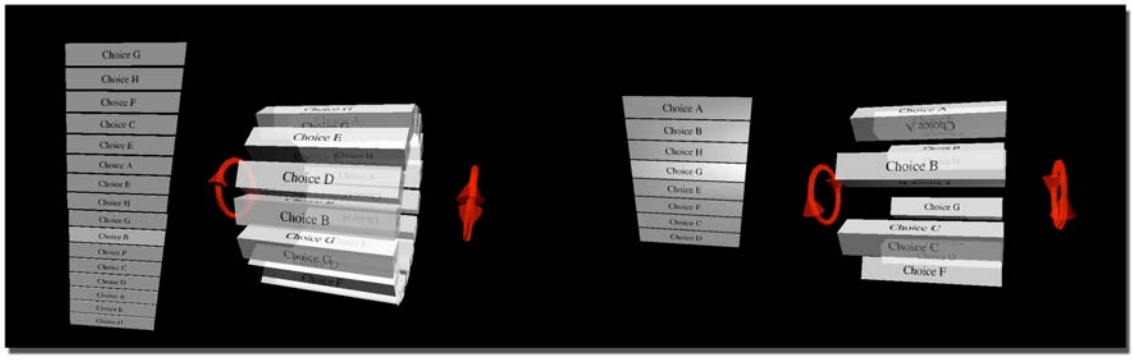



**Figure 6.30: Drums efficient use of screen space.**

The multilevel drum (using Russian doll principle outlined above) simply replaces a single drum item with a full (miniature) drum contained within a partially transparent drum item/container (see Figure 6.31). When moved over the surrounding shell/container

[27] Note that when viewed from the side, in a simple rectangular form (as shown), the space available for text is limited, this can be addressed to some extent, by using interactive expansion however the side view will always be slightly less effective.

becomes transparent allowing the user to see the sub drum in more detail. If selected the sub drum rotates to the front whilst enlarging in size (i.e. becomes the same size as the parent drum) and pushes the parent drum back in space thus making the sub drum the primary selection tool. When the user moves off the sub drum it reduces back to being inside its containing drum item in the main/parent drum.

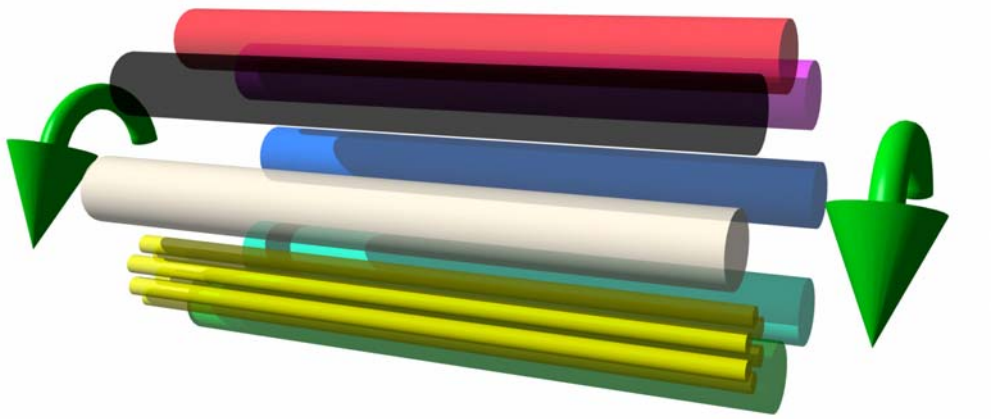


**Figure 6.31: Sub-drum (yellow) in a drum selector component.**

## The Implementation

As with all components in this research the drum component is constructed using the builder tool to construct a set of reusable elements that can be applied in a range of applications. The construction of the generic single level drum is used as the example for how the builder tool functions and as such its implementation is fully outlined in the *"Example Application of the 3D Interface Builder"* section earlier in the thesis.

Aside from the basic construction there were several key implementation issues in the creation of the drum component. The most challenging issue revolves around the effective presentation of two dimensional text based texture maps. Text by its very nature is written in a certain orientation (i.e. left to right, top to bottom), however if viewed from multiple angles the nature of what is left to right and what is top to bottom becomes confused. One option for dealing with this challenge is to use rectangular prisms for each drum-item and have them constantly rotate on their axis (i.e. so that the sides spin around and show differing faces). By mapping text in differing orientations onto the individual faces all options can be covered. Interestingly this technique could also be used to provide multiple languages (or other related information) for the drum items. Using rectangular prisms in this manner provides a functional benefit however a range of possible shapes could be used for the drum-items (i.e. ranging from rectangular prisms to cylinders, pie slices or others). The drum itself could be oriented either sideways (as in diagrams above or upright, which may be useful for vertical text such as is used in many Asian languages).

Implementing the sub-drum functionality introduces some new challenges, especially in terms of determining where to push the parent drum (i.e. in order to push it back) when the user activates a sub-drum. Knowing that the component can be used/viewed from

many angles, the ideal system would push the parent back into a location where it did not block any viewers location. Unfortunately this is not possible, however the system does apply some basic heuristics to minimize this possibility. Knowing that it is most likely that the user is manipulating a sub drum that is nearest to his/her viewing location makes it possible to simply push the parent drum in the direction of a line going from the sub-drum items container item towards the centre of the parent drum. Then move the sub-drum into the old position of the parent drum. There is no perfect solution to this problem as there is no location in which another user could not be located, however this system provides a system that is likely to generate the best possible result for the user who is interacting with the component.

## Summary

The drum component provides a generic reusable item which can be applied to the presentation of small to medium sized sets of options (ranging from 1 to 30). These sets can incorporate sub sets through the drums ability to replace drum-item elements with complete drums, thus making a drum inside a drum. Essentially it is a simple small set selection tool to handle tasks similar to those that are effectively handled in two dimensional systems by the menu component. It has the unique feature of being effective from all viewing perspectives and also makes more efficient use of viewing space than its 2D equivalents.

Contributing through its innovative method for the presentation of small sets of options, this component also identifies methods for presenting multiple versions of an option (eg. differing orientations and languages) within a single selection component. However its primary contribution is its presentation of a set of choices using a three dimensional rotating layout of options, this method can effectively display more items using the same amount of screen space than its competitors and in doing so provides a simple to use , yet powerful selection tool.[28]

## Discussion

The drum offers a rotating set of options and although this presentation enables more items to be presented in the same amount of screen space there is a risk that the rotational nature of the component is slower to use than its competitors (i.e. sometimes a user may be forced to wait while the desired option works its way to the front). This becomes more of an issue the larger the set of items, however given that the component is intended for use with small to medium sized sets it is envisaged that large sets involving long rotation periods would not be suited to this component. For the target small/medium sized set task the rotation times should be small and as such not cause significant delays.

A key question that arises with regard to the drum is why is it any better than a scrolling list? The answer to this is in several areas, starting with the fact that it is three dimensional and the use of the depth of the cylinder gives the user a sense of the presence of the other options (i.e. those at the back). This sense of the existence of more options is

---

[28] For results from the user trials comparing the drum to other systems, see the results section later in this thesis.

not there with a Supercube or scrolling list style interface. In those systems the user needs to understand that there are additional items and they materialize when scrolled to. The drum on the other hand taps the users understanding of 3D space by utilizing the idea of bringing items around from the back. This is a concept that is broadly better understood than that of items that just disappear on/off a list.

Essentially the drum provides an effective mechanism for presenting these small sets, in certain circumstances other mechanisms are more suited, however in the mainstream (eg. 6-20 choices) the drum represents an effective method of presenting a selection set to the user within a 3D user interface.

### *6.4.6 The Simpler Components*

This section covers a mixture of the less complex components (eg. simple buttons) as well as those developed for comparison purposes (eg. 2D menu, slider, window etc). Each of these components is implemented to achieve a certain task in the system, whether it be a simple setting device like the simple buttons or a device based on a 2D equivalent (eg. slider etc). Each component is implemented in essentially the same form as it is in 2D systems (this allows them to be used for comparative analysis) with minor variations as described below.

#### Controller Buttons

The button is a very simple interactive component and is a fundamental piece of the mainstream two dimensional interfaces widely in use today. Essentially it simply fills an area of the screen with a click sensitive rectangle. This rectangle when clicked initiates an action in the interface. This very simple mechanism is easily transferred into a three dimensional interface. By taking any three dimensional structure (a simple box is effective) and making it click sensitive this enables the developer to place these "clickable boxes"/buttons anywhere inside the 3D environment.


**Figure 6.32: 2D button example**

As the diagram on the right shows, GUI buttons generally use highlighting techniques (i.e. the blue button) to indicate the currently selected option. This technique can also be directly transferred into 3D interfaces by simply setting the buttons surface to be the highlight colour/texture when it is selected.

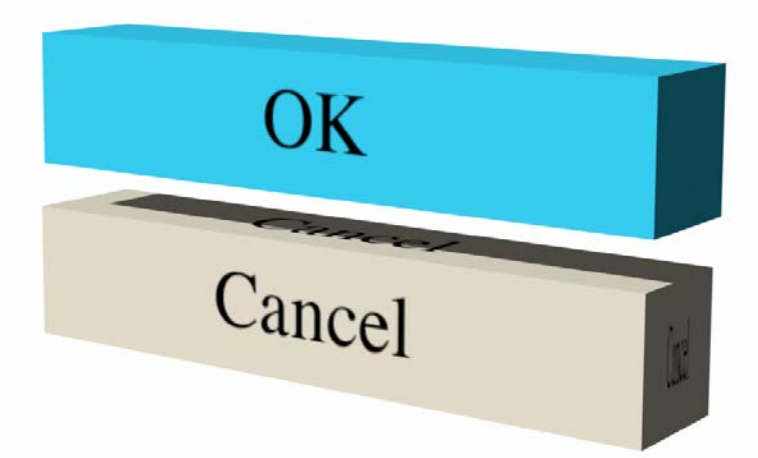



**Figure 6.33: 3D buttons (highlight on OK)**

One issue that 3D space introduces is the problem of texture orientation relative to the users viewing location. In a 2D system the text on a button is always upright and square on to the users view, however in a 3D system the user can approach a 3D button from any angle. For single user systems this can be overcome by orienting the button relative to the users view, however such a solution is ineffective in multi-user, multi-viewing angle systems. One solution is to have the button rotate in space (i.e. spinning on its axis so that

it presents differing angles to the user over time). Another is to use textures that are effective from all angles (i.e. rather than a single one directional “OK” have several mirrored versions of “OK” on the texture map itself, to present better angles. Although the mirroring technique is effective it creates buttons that appear complex (i.e. multiple text in different orientations), with smaller less clear text which may confuse users. For this reason the 3D buttons implemented use the spinning in place method to effectively present their textures.

From the developers perspective the simple re-usable “3D button” component enables the rapid development of a button based interactive system within a 3D interface. However the developer of a 3D interface must deal with several additional issues which are not faced by a 2D GUI developer. In a 2D system the developer needs to design and layout his buttons so that they do not overlap in 2D space. The same is true for a 3D interface, where the developer needs to lay out buttons so that their volumes do not intersect. In addition to the basic principle of avoiding overlap the 3D developer must also consider whether buttons will occlude others when viewed from certain angles. For example in the diagram above, the buttons are laid out nicely for the view shown however if viewed from above the OK button would block out the Cancel button and thus the overall system be ineffective. To lay out a system of buttons in a 3D space is quite a challenge. this issue is addressed in an automated form by more complex interface elements (eg. in 3D the Drum component described by this research). The basic button does not support any particular arrangement patterns and the layout of buttons remains under the control of the interface developer.

The 3D button component is used in this research in a range of areas from controlling user interaction (eg. click to start/continue etc), forming the base element of other more complex systems (eg. drum consists of an automated set of buttons) to the simple use as a comparative system for analysis purposes.

***Summary***

The 3D button component takes the principles that have been so effectively demonstrated by buttons in the 2D GUI interface and extends those to function smoothly in a 3D user interface. This component/interface contributes to the research through its provision of a reusable, general purpose interface component which can be applied to bring the functionality of the essentially 2D GUI button component, in the form of the “3D button”, into a fully 3D interface. It also contributes by identifying some of the key problems associated with moving an existing interactive component between the 2D GUI systems and the 3D systems and in so identifies key areas (i.e. button layout, texture orientation) that a 3D interface developer must address to create an effective system using button based components.

## Control Devices

The control devices are the interface components specifically developed to provide mechanisms to control the actions of other more complex components. Developed through a necessity to provide simple functional control each of these components is generally tightly linked to its associated larger component. Designed to provide basic functionality these controllers, generally are quite simple in function yet enable the larger components to achieve their more complex tasks smoothly and effectively.

### *Spinner Devices*

The spinner device is essentially a mechanism for spinning or rotating an associated item (or set of items) in the world. Presented as a permanent spinning ring with rotating arrows, this device is primarily intended to achieve two tasks:

- Attract the users attention to its function as the controller that spins the associated item/set
- Enable the user to spin the associated item/set

To enable the functional side of the device (i.e. its ability to spin the associated item/set), the spinner device is sensitive to user focus (i.e. if the user moves over it with the mouse/pointer); when moved over the spinner device rotates the associated set in place thus generating the spin effect. For an example of the use/implementation of the spinner device (i.e. in the drum component) see the earlier section *"Example Application of the 3D Interface Builder"* of this thesis.

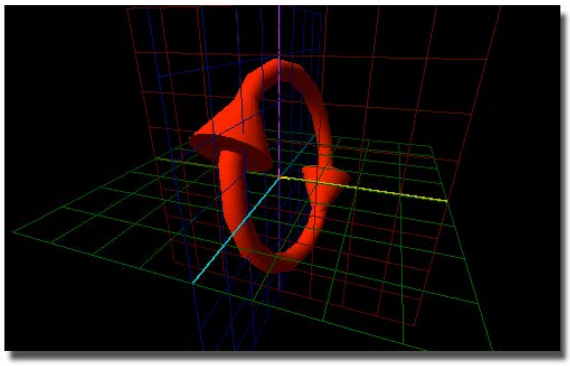


**Figure 6.34: The Spinner Device**

Attracting the users attention to the device and indicating its function is more challenging. To achieve this the spinner device is (by default, this can be set by developer) brightly coloured and in constant motion within the interface. This motion involves spinning about its own central axis so that the cone shaped arrows are seen to move around the ring (as shown in the diagram above). This spinning nature makes the device stand out in the users view and through its rotational action indicates its function to spin the associated set/item.

A very specific functional component the spinner device provides its spinning functions primarily for use in controlling the rotation of the drum component, however this generic spinning device could be used to rotate any linked item and as such provides a general purpose component for controlling the rotation of items in a 3D world.

### *Flow Control Devices*

Several of the complex interface systems developed in this research utilize the principle of a flowing set of data (i.e. flow component, tunnel, circulatory system). To implement

these systems there is a need to provide simple components to control the flowing data that they manipulate. As described previously in the *"Flow Component"* section of this thesis these devices are very simple in functionality and designed to be based on the widely known video control systems. As with all control devices it is critical that the user quickly realizes that this device is the means to controlling the larger system and for this reason the decision to use a well known flow control system as a basis for the device was taken.

In addition to the "flow component" control devices, there are also several other flow control elements used in this research. These are primarily part of the larger circulatory system interface (described later) and involve the use of items, much like road signs to enable users to control the movement of the flow relative to their location. For example a right turn sign in a flowing system is used to indicate that a user can click it to branch into the right branch of the flowing system.

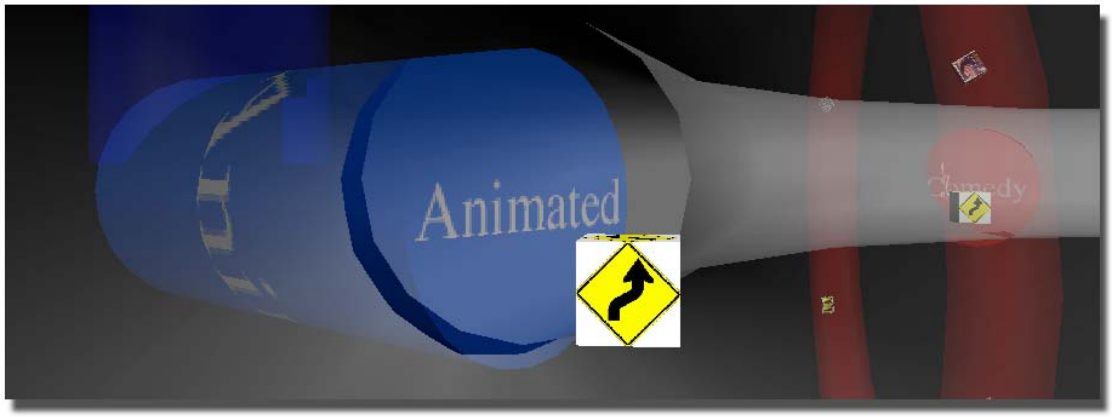



**Figure 6.35: Road signs for embedded flow navigation**

This concept is relatively complex as it requires an understanding of both the nature of the flow and the ability to control its direction, however the use of widely accepted international road signs brings a known system of flow into the interface and simplifies the users understanding of the concept. These road-sign based components are essentially just simple buttons with a particular form of texture map, however they serve a particular role in controlling flow very effectively.

### *Summary*

The control device components are elements built more out of necessity than any other overall design goal. They provide critical control features to the larger and more complex systems and as such are vital elements of any complete 3D user interface. Unlike other components there is a strong need for these elements to indicate their function to the user through their appearance and actions, as it is through these devices that the larger systems are controlled. Utilizing well known international standards as well as basic techniques such as movement and colour enables these devices to grab the users attention and focus it on these control mechanisms.

## The Menus

The concept of a menu is one that is well known to any two dimensional interface developer and is one of the cornerstones on the two dimensional GUI interface. Although the menu has become tightly linked to the 2D interface the initial concept of the menu is based on a real world (i.e. 3D) system. The menu that is used in restaurants, although planar and 2D in implementation the menu functions very effectively in the real three dimensional world. For this reason and for use in comparative analysis with other small set selection interfaces a collection of menu inspired interfaces have been developed to provide 3D developers with an easily accessible component for implementing menus in a 3D interface.

### *The Flat Menu*

This components essential purpose is to provide the well known, planar 2D menu that is widely used in 2D systems in a form that can exist at any location in a 3D space. In addition this component provides an excellent component for comparison with the 3D drum component described earlier. It is basically a 2D menu inserted in its two dimensional form into the three dimensional world. As you would expect its primary task is to enable selection from small to medium sized sets.

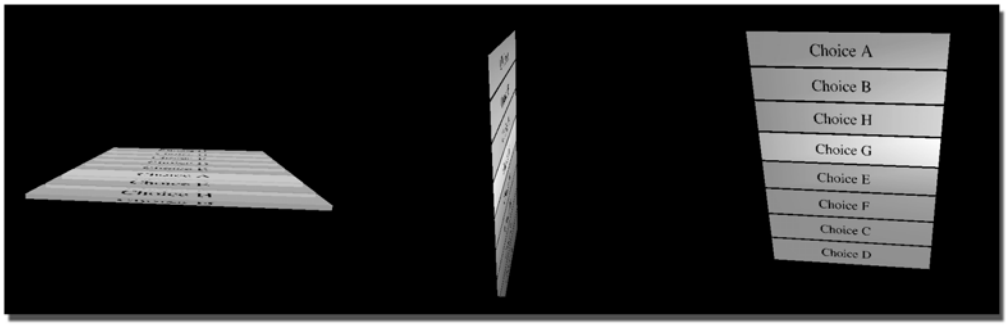



**Figure 6.36: The flat menu in a 3D space**

As the diagram above demonstrates the flat menu, although extremely effective in 2D suffers when viewed from any angle other than square on. This is a limiting factor for the flat menu, however if applied only in single user circumstances, where the menu can be oriented relative to the users view (eg. image on right above) this issue becomes less relevant (see [Larson 00] for more detail).

As with other components developed in this research the flat menu is constructed to provide a generic re-usable component for application in a range of potential interfaces. To achieve this the flat menu is broken into two sub elements. First is the menu item, this is simply a planar button with an associated action (on isOver highlight and on Click do action) and texture. The second is the flat menu group, this is a group object containing the set of child menu items. Thus a developer can build a specific menu by simply creating a set of menu items and their actions and adding them as children to the flat menu component.

Although simple to construct the fundamentally 2D nature of the flat menu makes it unsuited to use in multi-user systems and also for use at any fixed location/orientation in 3D space. However when used for single user systems and oriented relative to the viewers location this system functions as effectively as its 2D equivalent.

### *The Box Menu*

The box menu is a simple extension of the flat menu. It uses boxes (i.e. with depth) in place of the flat planes used in the previous component. Functioning in the same way as the flat menu the box version is also implemented using constructive sub components, the difference being in the menu item component which is a box rather than a rectangle. This additional depth solves one of the two key viewing problems associated with a flat menu in a 3D environment. This is that the box menu when viewed from either of the front or side views effectively displays its options to the user (as compared to the flat menu which is only effective from the front, see Figure 6.37).

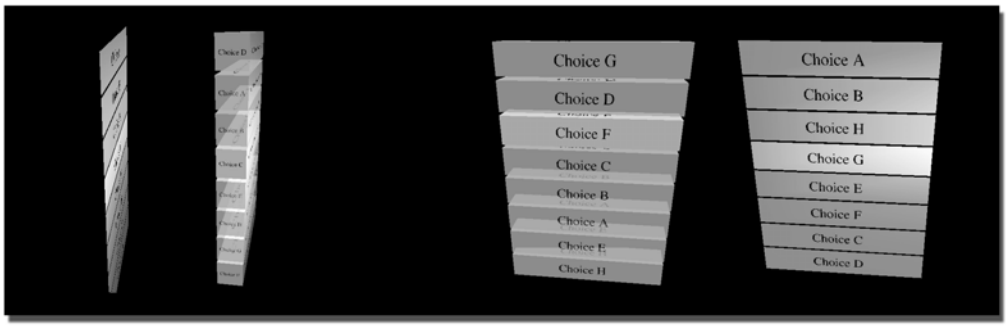



**Figure 6.37: Flat menus (outside) vs. Box menus (inside)**

Although the box menu solves the side view issues it only slightly improves the situation with reference to the menu when viewed from above. This is not due to a lack of depth in the items but instead is caused by the fact that the top item blocks out the view of those under it. To help address this issue partial transparency was implemented (i.e. all menu-boxes are 50% transparent, so that when the user moves over one it becomes solid and thus the user can see the item through the top of those sitting above it). This partial transparency cannot truly provide the desired see through effect as it is almost impossible to determine which item is selected when looking through a stack of 50% present items above it.


**Figure 6.38: Flat menu and Box menus from above**

Despite the top view limitations the box menu makes better use of the depth offered by a 3D environment and hence should prove more effective in presenting its small data sets to the user that its planar equivalent (for comparative results of these systems see the results section for the drum component later in this thesis). Despite the difference in depth, in essence the box menu remains a subtle three dimensional variant on the well known 2D menu, the only key difference being the use of boxes rather than planes for the menu-items in the set. As is the case for the flat menu this component was developed to provide both a simple

reusable component for inserting the essentially 2D box menu into a 3D world and for comparison purposes with the drum and other 3D components aimed at this small set selection task.

***Summary***

The menu based components, although essentially 2D in functionality can be effectively utilized inside 3D environments. In order to provide an easy to use mechanism to construct such menu systems these components were constructed. By providing fully functional menu elements these components enable developers to rapidly create menu based systems within 3D environments by simply reusing these existing generic systems. In addition to this basic functionality the box menu also introduces several concepts such as partial transparency in an attempt to improve the clearly problematic viewing from above issues. Aside from these issues the core purpose for implementing the menu based systems is to provide a system against which many of the new and innovative 3D interface components (eg. the drum) can be compared.

## The Line Slider

Incorporating interface elements for setting values that fall within a fixed range represents a fundamental feature of almost all user interfaces. In two dimensional systems the slider is one of the most popular methods for setting this type of range data.

**Figure 6.39: 2D GUI slider.**

However in three dimensional interfaces components for setting this type of value are largely unavailable. A classic example of this is in observing most visualization systems, where the complex data being visualized is shown in the form of a colourful immersive 3D presentation yet the controls that enable users to manipulate the visualization are simply 2D sliders overlayed over the immersive 3D world. Although systems of this type function effectively they do little to maintain the immersive sense for the user.

The need for a fully 3D component to set this type of value is addressed in two areas in this research, firstly in the 3D Slider component which demonstrates several new interface concepts for handling this task. The second method is the simple implementation of the line slider described here.

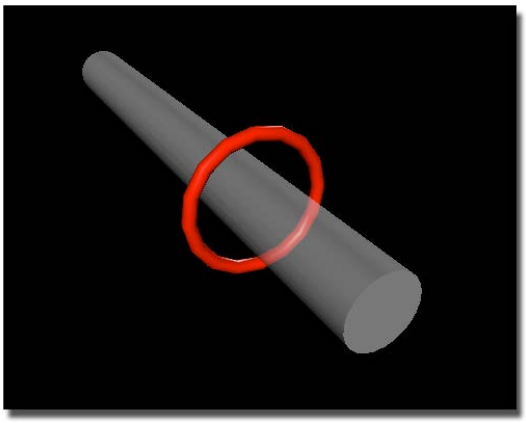


**Figure 6.40: Screenshot of line slider**

Essentially the line slider takes the mechanism used in a standard 2D GUI slider component and extends that concept into a simple three dimensional version.

The line slider is structured in the form of a simple cylindrical tube (to represent the range of values) and a ring which surrounds that tube (to represent the current value). This ring object is sensitive to dragging and as such can be dragged along the cylindrical tube, thus setting/changing the value. In addition the tube itself is sensitive to clicks and when clicked the ring moves to the location on the tube that was clicked. This implementation is essentially identical to its two dimensional counterpart.

The implementation of this component is, like all of the components in this research based on creating a generic reusable element and as such the line slider is constructed in a form that provides the interactive dragging and value setting features, these features can then be applied to specific cases by the final system developer.

The line slider is not intended to be a particularly three dimensional component, in fact like many of the 2D based components in this section it does not function particularly well when used outside of a “square-on” or 2D system. The objective with this component is to see if an effective 2D component can be shifted (with little or no change) into a fully 3D interface.

***Summary***

The line slider represents a 3D version of a relatively complex 2D interface component (i.e. the slider), which carries out a fundamental and very important value setting task in 2D systems. Essentially the line slider makes few changes to its 2D parent and this makes it an excellent test case for comparison with the new 3D components described in this research. In addition by constructing it in a generic reusable form this provides a mechanism for those developers who wish to use a slider within a 3D interface to quickly and easily implement those systems.

## The 2D Window Tree

The tree interface is essentially based on the windowing/desktop environment commonly used in two dimensional GUI interfaces. Its primary purpose in this research is to provide a comparison system for the presentation of structured (i.e. hierarchical) sets of data. That being data that is categorized and displayed based on some organized scheme. Taking the window/desktop based environment into 3D space introduces several issues, including:

- The concept of window depth (i.e. as the user can be anywhere in 3D space the concept of “front” becomes very view specific).
- Unconstrained working space (i.e. the user can place items out of view).
- The users orientation to the tree (i.e. the user can be oriented to view tree from any angle, not just square on).

To overcome these issues the tree interface uses 3D objects to represent items and grouping components in space (thus enabling the user to view them from any angle). The depth and working space issues are not specifically dealt with by this component. However it would be possible to implement spinning motion to bring the items to the near point of the users view when selected (i.e. this is similar to the Cone Trees interface described in [Robertson 91]). Equally constraining the range into which “groups” and “sub-groups” can be placed would effectively contain the unconstrained working space issue.

The 2D tree component implemented consists of several key features. The first is the “basic item component”, these represent files or simple items in the world. For the purposes of this research “basic items” are simple 3D boxes each with unique textures. Beyond the basic items is the first of the grouping components the “window box”.

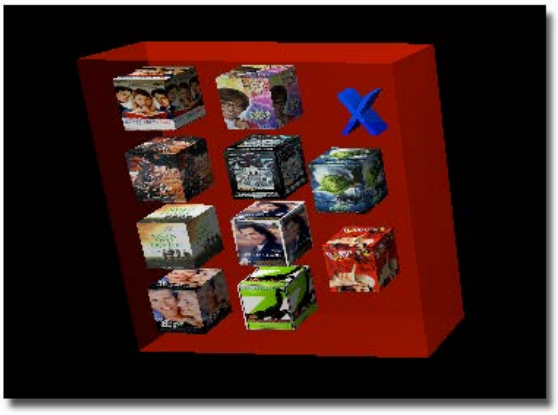


**Figure 6.41: Screenshot of “window box” component.**

The window box essentially has similar features to the well established GUI window. It is fundamentally a grouping component to enable the users to arrange groups and sub groups of data. Constructed as a 3D, partially transparent open topped box which can hold a set of other items (they can be either “basic items” or other window boxes). These “window boxes” can be opened (by selecting them in a parent “window box” or closed (by clicking the colour cycling 3D “X” in the upper corner).

This basic grouping component enables a tree of “window boxes” to be created to fill a 3D space. Each box containing both sub “window boxes” and “basic items”.


**Figure 6.42: Tree of "window boxes" and items.**

### *Implementation*

Essentially the tree interface consists of the two elements, those being the "basic item" and the grouping component the "window box". Intended to provide a window-like interface in 3D the tree interface is implemented through the construction of the two key parts that make it up (i.e. the "window box" and the "basic item"). The more complex "tree" simply consists of an hierarchically arranged set of these basic pieces.

Each of the sub components is constructed in a generic, re-usable form to enable it to be taken and applied in future interfaces. For this reason the functionality is kept to the basic minimum required. To achieve this the basic item (i.e. an element within a window box) is simply a geometric cube with a unique texture map and an attached action to undertake when clicked. The "window box" is more complicated because it has both a simple open topped box geometric structure, some basic interactions (i.e. close on X click) and also a set of child items. Each of these features is implemented through the builder tool and involves the construction of the geometric shape, the setting of the interactions and actions to take and the grouping of the child items below the overall "window box" structure. In addition each window box can have the "close X" element either included or excluded as desired (this enables developers to create window boxes that can remain fixed and can't be removed or make them closable). Overall the "window box" is simply an empty geometric box with an optional close mechanism and a set of child items. The tree interface itself is actually made up of a set of these very simple pieces, however as is the case for its 2D variant (i.e. windowing system) it is the arrangement and use of these pieces that makes it a more complex system.

***Summary***

The tree interface takes the principles applied in the 2D GUI desktop/window system and applies them directly into 3D space. Implementing windows as 3D “window boxes” and the desktop as the full 3D environment space into which these boxes and sub elements can be placed.

This component/interface contributes to the research through its provision of a reusable, general purpose interface component which can be applied to bring the functionality of the essentially 2D GUI windowing system, in the form of the “window box”, into a fully 3D interface. In addition it contributes through comparative analysis with other methods of presenting structured data (eg. Circulatory system). In addition it also contributes by identifying some of the key problems associated with moving interaction techniques between the 2D GUI systems and the 3D systems (eg. unconstrained space, orientation of items).

## Summary of Simple Component Set

The components outlined in this section in general are simple in their functionality, yet they provide critical features that enable the other more complex 3D interfaces described in this research to function. The basic buttons offer a simple yet invaluable method of enabling the user to start, stop and control the various actions of an interface. The control devices enable users to interactively control the large scale interfaces like the flow component and the circulatory system. The menus, sliders and window boxes offer versions of these popular 2D components in a form that can be used within an interactive 3D interface.

Each of these components is critical in its task, yet their functionality is simple. In general the components described here are usually unnoticed inside larger systems, yet without them those systems would be ineffective and when implemented as reusable elements they enable developers to quickly achieve complex interactive tasks.

## Discussion

The key argument against these essentially 2D interfaces and components (menu, slider, window boxes) is to argue why implement them at all. They are essentially 2D and will clearly not move smoothly into a 3D environment (i.e. all are designed to work only from a “square on” perspective and as such will struggle to perform effectively in a fully 3D environment). These components are implemented primarily for the purposes of comparison with the new and innovative 3D interfaces developed in this research, however whilst providing these comparative systems it is also an opportunity to determine if two dimensional components can be altered (albeit slightly) to become functionally useful in 3D interfaces.

### *6.4.7 The Circulatory System Interface*

The circulatory system interface/component represents a new method for the presentation of large structured sets of data. This interface is inspired by the need to provide an effective interface (in three dimensional space) for the simple task of selecting or locating a movie from within a large collection or library of movies (be it at a video store in the real world or through an online collection). It is designed to handle the task of selecting from a large structured set of data (i.e. where the items are grouped into sections and subsections based on a particular grouping strategy) and as such can be applied in a number of areas from mainstream file systems to specific movie, music, document and other libraries. The movie library example is a common real world example of selection from a large structured set in that movie libraries are usually categorized, into sections and subsections based on systems such as film genres and ratings. For this reason the movie selection task is the example used below.

The task of locating an item from within such structured data sets is not a new challenge and there are several systems that are widely used for this task today, including the GUI window system and various other search techniques. Obviously the graphical user interface with its desktop, files, folders and window based interface is the most commonly adopted method, however this system is fundamentally two dimensional in nature.[29] The challenge for the circulatory system designed in this project is to utilize the three dimensional space to effectively handle this task. The next most common and perhaps the most useful method applied to locating items within structured data sets is the search approach. The importance and power of search tools has grown significantly with the development of the Internet and the web. With so much information available existing techniques (involving manually navigating through structure) for browsing through the data are less effective and as such targeted searches are the best method of obtaining the desired information. Search tools and agents enable users to quickly locate desired items without needing to navigate through the structure of the library. This is very true in the movie library example being applied here, because a user who goes to a movie library knowing the specific movie (eg. titles, directors, actors etc) that he/she is seeking will be able to locate that item most quickly by using an effective search tool. This targeted approach to locating items in the set is already effectively handled by a range of search tools.

In the case where the user is not sure what they are looking for and wants to browse a collection, hoping something of interest will leap out at him/her. This browsing technique is usually the method through which people select a range of real world items, ranging from shopping for groceries to movies. Interestingly this task is something which most existing interfaces do not focus on. A useful 3D interface system for enabling this browsing activity is worthwhile and it is this browsing task that the circulatory system described here primarily targets.

---

[29] See previous section "*The 2D Window Tree*" for a three dimensional implementation of this mainstream system.

In the (relatively common) browsing scenario the user does not want to have to actively rummage through the shelves looking at every option. Ideally the full set could present itself to the user (i.e. so he/she can take a passive viewing role). This active presentation of data to a passive viewing user is suited to the 3D interface as clearly demonstrated by the flow component described in the earlier section. Essentially such a system enables the user to observe and make selections from within the set as it moves through and past him/her.

**Figure 6.43: The flow interface for unstructured data.**

The flow component works with an unstructured set of data and as such the data set can simply flow along the axis towards, through and past the user, making the most effective use of the depth of the 3D space. Structured data however has categories and sub-categories making this simple presentation as a single set flowing on a single axis impossible. The challenge with structured data is to present the categories and sub categories each as sets that flow past the user (in a similar way to the flow component) and yet also present the overall set as a whole. To achieve this the circulatory system interface is inspired by two key real world systems.

The first is the human bodies cardiovascular system (i.e. the heart and blood vessels etc). In this system red blood cells flow around inside a broadly branching yet permanently contained system.

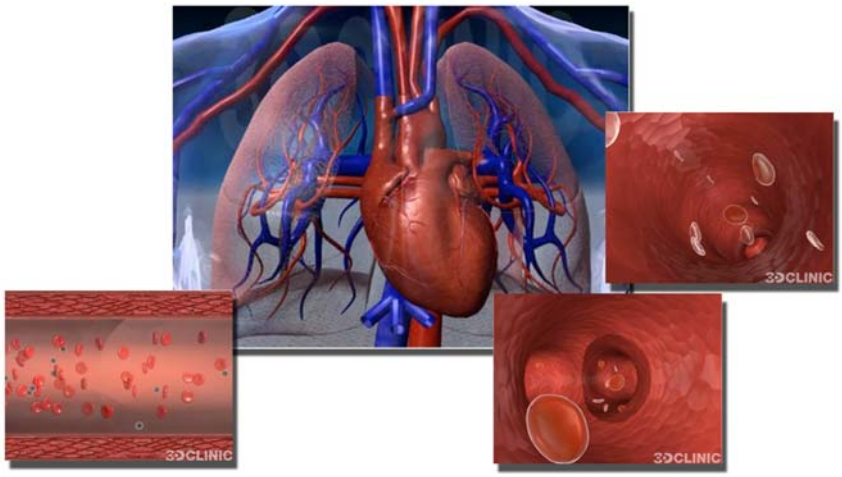



**Figure 6.44: Human circulatory system [Health 03].**

This flowing system enables each cell in the body to maintain its position in space yet receive oxygen and energy by being on a permanent flow of blood cells which provide those key elements. Essentially each cell sits on a stream and over time every red blood cell flows past. This flow of the set of items (i.e. red blood cells) past a fixed location (i.e. body cell) matches the type of presentation required for the flowing presentation of a structured set of data (eg. flow of a set of movies past a users location etc). However this still leaves the issue of how to represent and enable the user to navigate through the branching nature of the structured data. It does, however provide a basic conceptual outline for the interface.

The second real world item that inspired the circulatory system interface is Celtic art. In Celtic art, and particularly Celtic knots, there is a concept of a complete looping system. That is a loop with no beginning and no end, yet containing many potential branches and subsections as shown in the examples below.

**Figure 6.45: Example of Celtic knot work.**

Similar in layout to the human cardiovascular system these knot designs are a little simpler to work with and more clearly demonstrate the looping nature of the overall concept. Applying these real world systems, both of which incorporate flowing systems which branch into sections and subsections, is the objective of the circulatory system component. The fundamental design involves using loops of flowing items (where each loop represents a category) similar to the coloured loops shown in the Celtic symbol above on the left. Using this system the green loop (for example) could contain all movies in the category “Action”, while the other loops contain other categories of movies.

The basic action of the component is to have the movies in each category flow through the loop that contains them and thus when the user moves inside the space of the loop he/she is able to view the flowing set of items as it flows past and under his/her position. By using a partially transparent tube to surround each category, this enables the user to retain an understanding of their current state at any time and thus give a sense of their location in the current category. In order to enable the user to move between categories/loops the actual structure of the full circulatory system must also rotate in space, thus if the user is inside the green loop as the system rotates the join between the green loop and the yellow (or any other) loop slowly rotates into the view of the user and thus he/she is able to step into a separate category. This basic flowing circuits concept enables the structure of the circulatory system to flow past the user without him/her needing to actively move through that system (i.e. the system comes to the user).

## General Description

The circulatory system interface is based on (and attempts to closely match) the application of the conceptual real world systems (see above) and their application to the challenge of the movie library selection task. This interface essentially creates an overall system that mimics the actions of the real world human circulatory system, hence the name selected for the component. From a high level perspective the system is designed to present the user with a set of linked rings, each of which contains a set of items (eg. movies or links into other sub-rings) that are slowly moving through the tubes that define their ring.

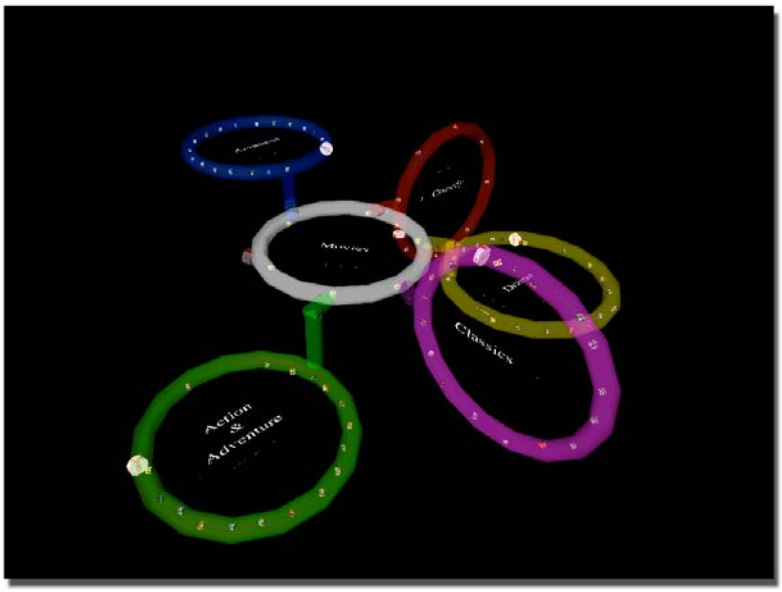



**Figure 6.46: Screenshot of a Circulatory System Interface**

The user is able to enter any of these rings (by simply clicking it). In doing so the users view moves from the external perspective shown in Figure 6.46 to being inside the selected ring, as shown on the right in Figure 6.47. It is from this internal ring view that the items appear to flow through and past the user (much like the flow component only that the flow moves in a circular path rather than a straight line).

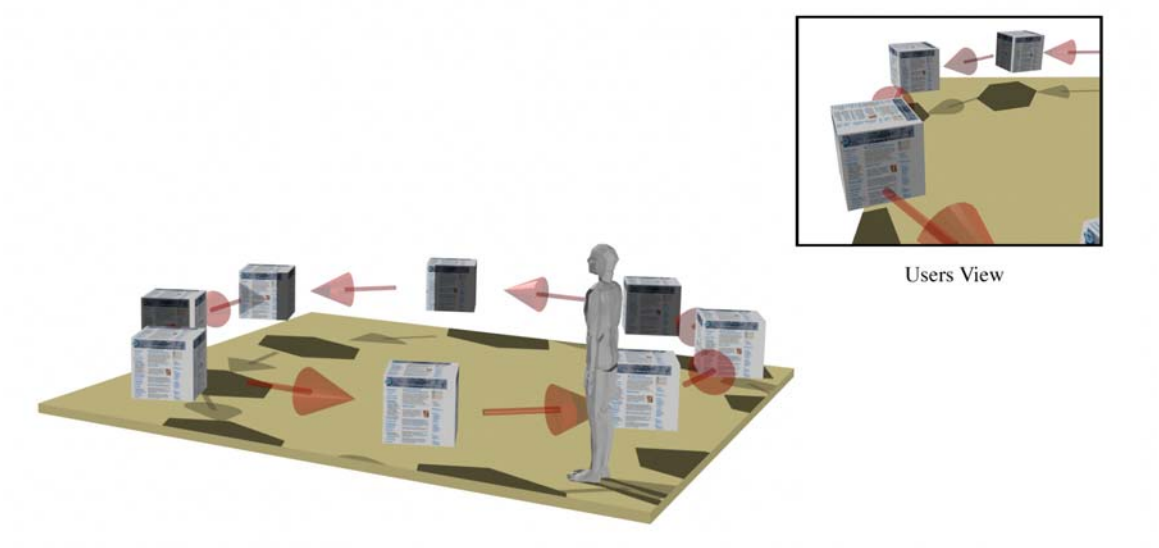



**Figure 6.47: Basic flow of ring of items.**

This circulatory system interface is designed to be able to be present (i.e. exist at a location) in a 3D space and not be dependent on a users viewing position to be effective

(note that this is different from the basic flow component). Hence a user can view the structure of the circulatory system from a range of perspectives (including inside the rings or outside) and still be able to effectively make use of its functionality. This enables the component to function in both single user and multi-user environments.

Designed and constructed using reusable elements (i.e. items, rings, branches) the circulatory system component can be applied to any set of structured data by simply creating the specific sets items and rings then using the joining and branching elements to bring them together into a complete “circulatory system”.

The key objective for this component is to provide a interface mechanism which can make use of the beneficial features of the 3D space for presenting structured sets of data. Particularly through utilizing the depth of the space to enable a flow of items to flow towards the user from a distance. This flowing system when combined with the branching loop oriented structural design enables users to view large structured sets without needing to navigate through the space, but rather having the data present itself to the user.

## The Implementation

The complexity of the information being presented by the circulatory system component makes the implementation of this component in a generic reusable form more challenging than many of the simpler components described in earlier sections. However by breaking the conceptual component (as described above) into its functioning pieces this enables the component to be described as a single large component which handles the complexity through a collection of sub-components.

As is the case for all components in this research project the generic circulatory system component is developed fully within the builder tool and is thus able to be reused for other specific cases. However in implementing this component there are several key issues that must be handled to enable the conceptual design described above to be created in a functioning and reusable form, including:

- The items (i.e. how to present the items in the set)
- The categories/rings (i.e. how to present the categories and sub categories)
- The tubes (i.e. how to contain items within categories, and is containment needed at all)
- The flow of items
- The branches (how to enable movement between rings)
- The automatic flow (how to enable traversal without taking away the users sense of control)
- User control (how to enable the user to control the flow, ring level etc)
- Presenting the overall set

### *Presenting the Items*

The set of data being represented has an existing underlying structure. In the case of the movie library this is categorized based on film genres, however any scheme to group and sub-group the set would be suitable. The items that make up the overall set, for the example case are essentially similar in type (i.e. all are movies), however this is not a necessity, for example the items could be of varying types, structures and shapes.


**Figure 6.48: A Movie Item**

The first key implementation issue involves selecting the physical form (i.e. 3D structure) to use for representing these items. In the movie example the items could be represented as flat 2D planes or as 3D cubes. The benefits of using 3D cubes over 2D planes (i.e. the depth is more effective for multi-angle viewing) are discussed in the *Flow Component* section earlier in this thesis, and it is for the same reasons that 3D cubes are used in the circulatory systems representation of movie items. Given that movies are "moving pictures" there is also the potential to use an animated texture map on these cubes, thus enabling the cube to actually be playing the movie (or a sample from it). For performance reasons (i.e. libraries with hundreds, perhaps thousands of boxes with animated texture maps would have an unacceptable performance impact), this movie texture method was not employed, however it represents a possible extension for future versions.

### *Presenting the Set*

As described above the overall structure contains a series of rings, each of which represents a category within the structured data. These rings are container items into which individual movie and branching items can be added. Structurally each ring is made up of a collection of items laid out in a circular pattern. This set of ring items is then enclosed in a partially transparent wrapper tube (i.e. like a doughnut around the ring of items). This full set (i.e. rings of items) then spins in place (i.e. the items rotate around the rings central point) thus moving the individual items through the tube (as shown below) and generating a steady flowing system.

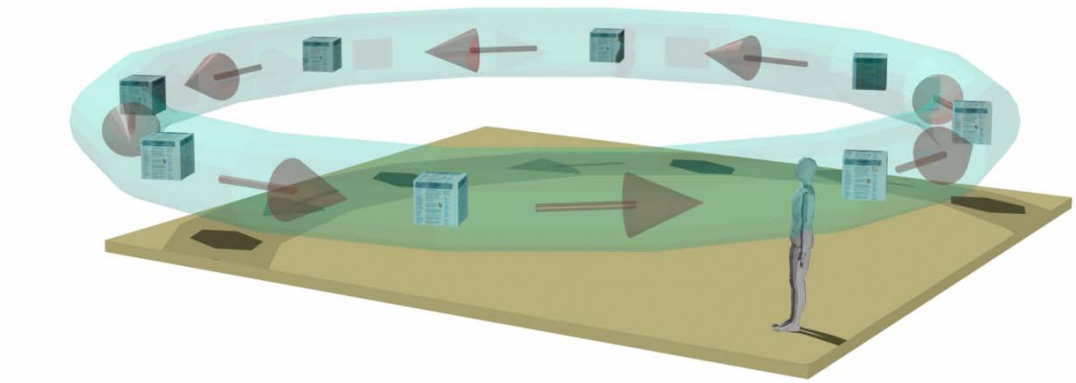


**Figure 6.49: The basic ring concept.**

This basic “ring” is an interface component in itself. It is a new re-usable method/mechanism for presenting a single set of data via a 3D component which can be located at any point in 3D space (i.e. the user can view the ring from any perspective and can enter it from any perspective). In many ways this is an extension of the “flow component” developed earlier in this research, with the new rotational structure and motion enabling entry and viewing from any perspective. This component also extends the “flow component” by providing both an external perspective (i.e. the ring viewed from the outside) and the internal perspective (i.e. ring viewed from the inside). With the internal version the users view is from a point inside the ring (at any fixed point) and from this perspective he/she will experience a flowing set of items which continually loops past his/her position.


**Figure 6.50: Screenshot of users view from inside a ring component.**

To create the ideal user view involves moving the user to a viewing location that is inside the ring but also looks in a direction that is a tangent to the inside of the inner ring, thus looking through the tube at the optimum viewing angle (as shown above).

As the diagram above shows the shape of the set/ring is a critical design and implementation issue, for example the diagram is an example of a users view from within a circular ring. As it shows, the items follow a path that does not allow a large “run in” range (i.e. they turn quickly and do not come at the user from a great depth), thus minimizing the benefit to be gained from the 3D depth of the environment. The “ring” could be implemented as a range of possible shapes, the only restricting factor is the need for the shape to form a loop (obviously more complex shapes and the associated motion paths will create more complex implementation issues). For that reason perhaps a circle that is simply extended in one (primary) direction would be a simple yet effective proposal. The circulatory system implemented in this research uses the fundamental circular style (as this enables the use of rotation as the form of motion and enables the simple layout of child items), however the possibility of more complex loops represents an area to be pursued in future versions.

The full circulatory system is fundamentally a collection of interlinked ring components with the items in each ring remaining contained inside their ring, thus retaining the categorization of the structured data. To enable these independent rings to function as a structured set the circulatory system uses a set of joiner pieces to enable the creation of paths between differing rings. Linked into an overall structure this “circulatory system” (made up of child rings) rotates around its own central parent ring (i.e. in the diagram below the coloured (child) rings rotate around the white (parent) ring).

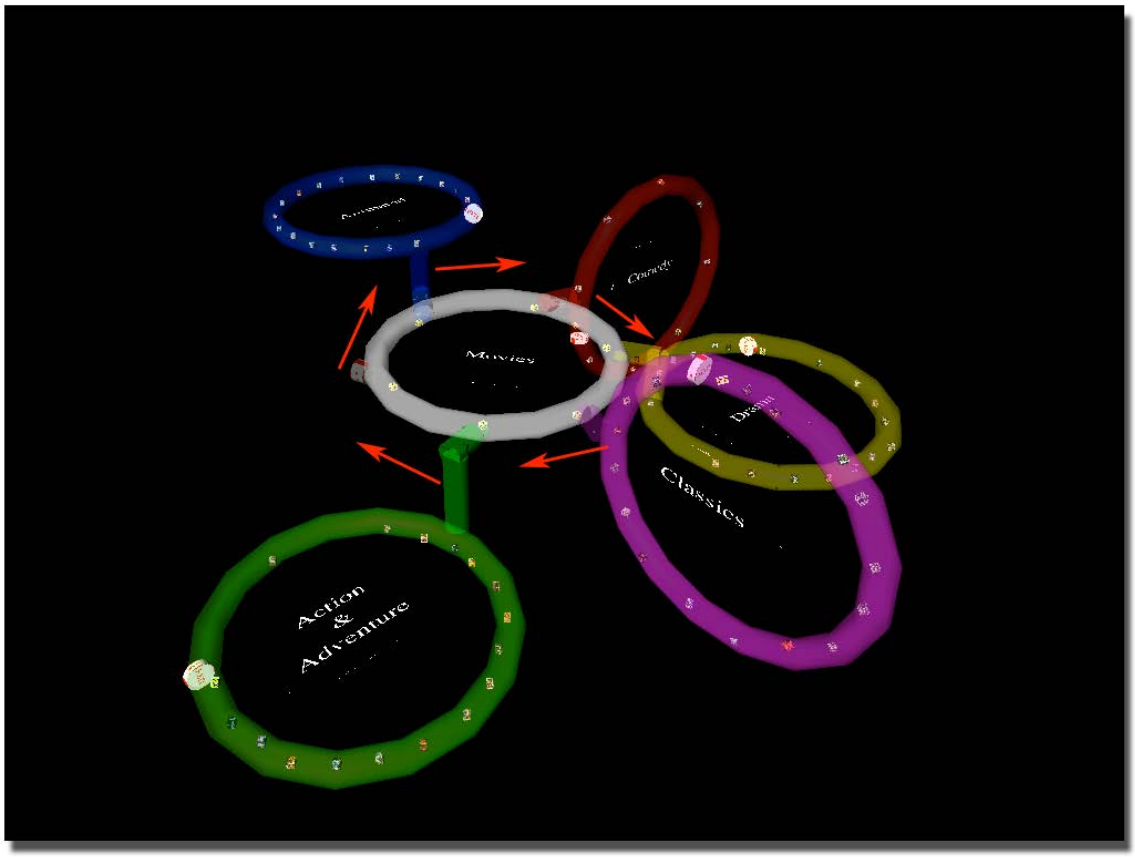



**Figure 6.51: Motion of the circulatory system as a whole.**

The system of having child rings rotate around their parent ring enables the user (whose location is fixed) to be exposed to a flow of items that includes the branches that join one ring to another. These branches are represented by the joiner pieces (i.e. small tubes that link one ring to another). As shown in the diagram above (which is a screenshot from the trial system) the green “Action & Adventure” ring branches off the main “Movies” ring via an L shaped tube. This tube also extends into the space inside the “Movies” tube and as such if the user were inside the movies tube this green branch entrance would cycle around past him/her and enable him/her to take that sub-sections branch if desired.

The overall layout of the circulatory system consists of the set of ring components (as required for the data to be represented) and the branch items between those rings. Arranging the rings in the available space, with the need to avoid collisions is a task that can be automated (see [Bell 01] for existing research into this “space management” concept). The current circulatory system does not undertake automatic space

management, instead it allows the developer to specify locations and branch type for the rings. Automated space management offers some interesting potential particularly in relation to maximizing the users ability to see as many choices as possible at any time and as such may be beneficial in the future, however giving the user the ability to lay out the shape of the rings (i.e. as is the case with the current system) also allows the interface to be personalized based on its particular task.

The fundamental design enables the user to view the overall structure from both an external perspective or from within. From both of these perspectives the motion of the structure (and content within it) ensures that all rings cycle over time and thus no matter where the user views the structure from, each section and sub section will either be visible or be cycling towards a visible location.

***Navigating the Structure***

The structure of the circulatory system component, through its own flowing motion provides a default system which enables the user to traverse the full set of information without any navigational tasks required. In addition to the flowing nature of the component there is a need to enable the user to control his/her navigation through that structure. This is a very important element because the user needs to feel, and be, in control of the system and it is through these navigational elements that control is provided. The key implementation issues relating to this navigation through the structure include:

- Providing a full system traversal method that allows the user control/steering while retaining the flowing nature (i.e. able to avoid unwanted branches but still keep the flow moving)
- Enabling the user to control the flow (i.e. able to start, stop and set the flow speed)
- Enabling the user to move between ring levels (i.e. internal to external jump)
- Providing the user with a sense of controlling the flow and path that he/she is shown.

The initial view a user will have of the circulatory system will be from an external standpoint and from this perspective the user can click any ring to enter at that level. Once inside the circulatory system navigation is fundamentally controlled by the branching behaviour of the user. Branching behaviour refers to the choices that users make when a branch (i.e. link into another ring) flows past them. At each ring intersection the user chooses to either enter or bypass a branch. Branches essentially exist at points where one ring intersects another. As these branching points flow past a user can choose to either flow into the branch (the default behaviour) or avoid the branch and continue in the current ring.

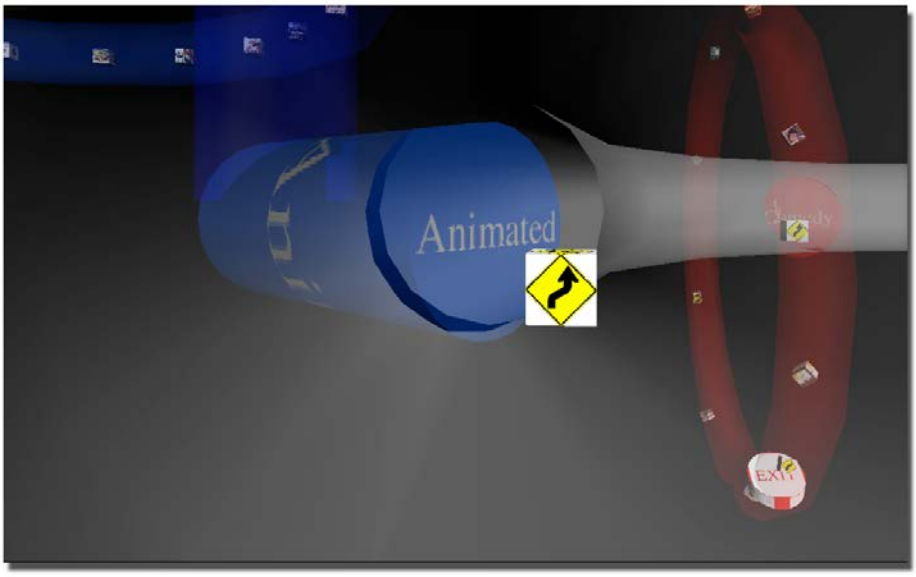



**Figure 6.52: Screenshot of a branch**

These branching points are implemented as simple cylindrical tube structures. These tubes flow past the user as elements of the set of items (i.e. they are simply branching items rather than movie items) and unless he/she chooses to avoid entering a branch then the default positioning of the branches takes the user down their tube into the new ring (implemented using a proximity sensor to detect when the user enters the branch and then take him/her into the new ring). This method creates a system where by default all branches are taken, and hence, over time the user takes every branch and traverses the entire structure.

Once inside the new ring the rotation of the items in that ring creates a flow of items past the user and at the end of the flow is the exit branch (which again is taken automatically unless avoided) which returns the user to the parent ring. This automated branch taking system can be controlled by the user through either moving his/her location or by clicking the bypass object (i.e. the yellow sign in the diagram above). In bypassing the branch the user will remain in the flow and continue on to the next item/branch. To implement these branching controls, standard international road signs were used to indicate flow options.

Standard controls, based on those outlined in the “flow component” were also used to enable the user to control the speed of the flow itself. These simple controls move with the user through the space and enable him/her to start, stop speed up and slow down the flow. In addition to these basic elements (from the flow component) the circulatory system also provides a jump level button (to go up to parent level) and an exit button (to allow the user to step to the external view of the circulatory system. The “level jumping” and “exit” tools are critical as some rings are large, contain many items and hence the embedded “exit” may be some distance away. If a user accidentally enters a ring these “move with user” tools enable him/her to quickly return to the parent ring. The “move with user” controls, in combination with the controls embedded within the circulatory systems structure provide the user with a readily accessible means of controlling this component.

The full circulatory system component is designed from a series of smaller pieces (i.e. items, branches and rings), with each new level (and the associated new component) becoming more complex and enabling the final “circulatory system” level to present large sets of structured data as a cohesive interlinked whole. From the 3D interface developers perspective, constructing a specific instance of a circulatory system involves constructing a series of ring components (i.e. one for each category of data) then using the branch component to link those together into a full circulatory system. Using a construction tool such as the builder tool described earlier in this research, in combination with the set of new components (i.e. item, ring and branch) this development process is a relatively simple challenge.

## Summary

The circulatory system is the most complex of the new components developed in this research. Essentially this interface provides a mechanism for using a flow of items to present a structured data set by providing a circulatory system interface which incorporates a flowing system of rings and sub-rings (representing the structure in the data set). Utilizing the advantages of three dimensional space this component identifies a method for presenting this structured set of data (i.e. such as libraries) in a flowing interactive format not seen previously in 3D interface research. Taking such structured data (eg. a movie collection) and using a unique traversal technique this component allows the user to simply sit and browse as the movie set (retaining its internal structure) flows past his/her viewing location. Additional flow controls enable a more targeted approach where the user can avoid the default flow to seek a particular loop or sub-loop of data.

Constructed in the form of a reusable generic component this “circulatory system” element can be applied in a range of different real world tasks ranging from selection of movies from a video library to selecting groceries from an online store. Introducing the concept of a permanently flowing, yet self contained in space, structure this component demonstrates that data sets need not be presented in a static form and can be presented as active information successfully (see results sections for more detail).

## Discussion

The circulatory system component identifies some interesting issues with regards to general 3D interface design. One of the most challenging issues is the choice of whether to include a tube surrounding each ring of data or not. There are advantages and disadvantages on both sides, for example the shell (in future the surrounding tube will be referred to as the shell) provides a mechanism for containing the ring of data. In addition it provides a sense of state to the user (i.e. when inside the red shell the user sees the red tube and hence has a sense of his/her location in the overall system). However the shell itself also blocks the users ability to click through it (i.e. when items flow from around the corner the user must wait until they clear the area that the shell blocks out before they can be selected. This same blocking issue exists from the external viewpoint where the user may want to directly select an item (without entering the internal view), but cannot because the shell is blocking the selection. The shell does offer one other key positive feature, that being in the task of detecting a user click and transferring a user from

external to internal viewing locations. The current version of the component implements the shell as an optional feature, hence the developer can choose to either include it or not. An assessment of the value of the shell (as indicated by user preference after using systems with and without) was undertaken in this research and the result (summarized in the results section for the circulatory system) indicate that users generally found versions without the shell more effective.

Another key design issue, and the source of some key implementation challenges for this component was the overall desire to never overtake control of the users viewpoint (i.e. let user navigate space, do not move him/her through it). In order to implement the flowing and branching features of the circulatory system the user would either have to be moved (i.e. move from ring to ring) or the whole circulatory system component would need to shift around him/her in relatively complex way to make it work. Early trials indicated that users found the moving structure to be less desirable than moving the user, hence the final system uses a set of small user moving actions (although both versions have been implemented to enable application in interfaces where control of the users view is not available). By briefly taking control of the users position and view (in order to jump into branches) the system is able to accurately position the user and ensure the internal view is most effective. Interestingly the users do not seem to be bothered by brief moments of controlled movement, perhaps this is due to the fact that they have selected the branch and expect to be taken there. Currently the movement is implemented as a simple jump from viewpoint to viewpoint, however there is potential to implement a smooth animation between the two (animated transitions of this type were demonstrably more effective in WIM systems such as [Pausch 95]).

In the process of development and testing for the circulatory system some unexpected discoveries were made including the identification of several new 3D interface design principles including:

- **User reorienting ability** - trials with this component have identified that most users are able to “upright” texture maps on items mentally. This shows that users can understand/identify items even when they are upside down[30]. Hence this identifies that although the upright presentation of texture maps is desirable it is not critical. This is a particularly valuable principle for application in multi-user environments where it is not possible for “upright” to be the same for all users.
- **Click through transparency** - interestingly most users expected to be able to click through semi-transparent objects in the world. Transparency itself was found to be very effective in providing structure and presence to items without having them block out regions of a users view. This indicates that transparency is effective and should be used, but where possible semi-transparent items should be able to be clicked through (or at least indicate that they are not, i.e. perhaps become solid when user moves over them). This principle has design implications in several areas, and is discussed in more detail in the “move with user” tools in the tool-belt component section later in this thesis.

[30] Not as effectively as for upright items, however a non upright item does not make it unrecognisable, in fact for most users there was surprisingly little distinction.

- **Active Presentation** - this component identified that interfaces that actively present their data can be effective for certain tasks. However where possible activity should be simple and consistent and excess movement should be minimised and kept to subtle variations.

***The Main Benefits & Risks***

A complex interface that moves the user around in space and actively undertakes flowing tasks without the users input, clearly exposes the main risk of removing the users sense of control. The sense of control is a critical issue and this interface pushes the boundaries in terms of actively presenting itself to the user. From the perspective of the data being presented this active interface is ideal for showing items to the inactive user, but is this activity taking the users sense of control? In general the users became more comfortable with the interface the more exposure they had to it. In most cases the initial sense of its running out of control gives way to a sense that it is flowing in a controlled consistent state that can be manipulated and used for the task at hand. For the user trials the flow controls were removed (i.e. to isolate the flowing presentation form at a particular speed etc.). Although the reinstatement of these controls provides the user with some control it is unlikely to remove the initial “out of control” reaction that occurs for some users. This is particularly prevalent in users with limited experience in 3D systems and hence with experience it is likely to reduce/disappear (this was evident even during the trials themselves). For more details on the performance of this system in the user trials see the results sections later in the thesis.

The flowing nature of the system can be reduced (i.e. the developer is able to control the amount of flow that occurs by manipulating the individual branches and rings - eg. limit to non branching etc.) to enable the user to navigate through the structure themselves. This flexibility enables the component to be used ranging from a static layout of rings, through partially flowing systems to the complete full traversing system described above. Hence the component is designed for application in a range of levels of “active presentation”, however in its primary task it is designed to maximize the use of the 3D flows for presenting the structured set.

Reducing the amount of flow and default branch taking would overcome the bulk of the “out of control” issues, however it would also remove the key benefits of a system that presents the data rather than a system that holds the data and waits for the user to navigate through it.

Essentially the primary objective of this interface is to provide an interface to enable the user to browse through the structured set. Clearly for more targeted searches other methods are more suited, however for browsing the ability to enter the circulatory system and have it flow all of the options in three dimensions through and past the user provides an innovative method for this passive browsing activity.

The circulatory system represents a new method of presenting structured sets of data in a way that allows the user to browse through the content without needing to actively seek any particular item. In today’s information environment there is a flood of information at our fingertips as never before. However the methods that have been used in the past to navigate, search and view such data are less and less appropriate in this large scale

information world. The circulatory system introduces the concept of information as a flow so that a user can simply enter the flow and observe information as it flows past. This more passive form of browsing through a structured set of data offers many benefits with regards to presenting large sets of data, however it also introduces some significant changes to the standard interfaces of today. The flowing interface clearly has enormous potential, yet also requires careful development to create a system where the user retains his/her desired level of control of the system.

### *6.4.8 The Full Environment Interface*

The concept of a three dimensional user interface involves the user having a presence (or view) into a three dimensional environment. To obtain the greatest benefit from three dimensional space the user is able to move that view within the 3D environment, thus enabling the user to move around in the 3D space. This basic principle applies in all versions of 3D interfaces whether they are displayed as fully immersive stereo environments (eg. virtual reality systems) or simply as 2D viewing windows into the 3D space.

In providing the user with this flexibility of movement, interface designers are faced with several new issues. In two dimensional systems the users entire working space is contained within the rectangle of the users desktop. As such developers can locate items in this rectangle and know that the user will be able to view and interact with them. In 3D space the working area is less constrained, in fact it is largely unconstrained, and as such it is difficult for a developer to locate items in space so that they are guaranteed to be in the users view at any particular point in time. This issue could be addressed by restricting the users movement/orientation or limiting the working region of the user (i.e. as in 2D systems the desktop space is usually mapped to the size of the users screen). Restricting the users movement would solve this problem but in doing so would fix the users location and orientation and thus remove many of the features that make a 3D interface valuable. The other option of limiting the users workspace to a fixed range also fails to address the problem. The key issue with limiting the workspace is that the user can simply reorient his/her view and the items that are placed in front from one orientation are actually behind the user from another.

The obvious solution to this issue is to not simply have a 3D space filled with items at fixed locations but to also provide the capability of attaching items to the user him/herself. Early virtual reality research [Fisher 89] identified the need for the user to have a sense of their presence in the 3D world. This challenge was addressed by providing a “virtual hand” to provide the user with an item that moved with him/her through space. Extending this concept to include not simply a representation of the user but to also incorporate tools and other items that a user can carry with him/her is the focus of this section.

## The Portal Component

Three dimensional space is a large and unstructured area. One of the key problems with many three dimensional systems (where the user is able to move in the space) is the fact that the users move into empty space and can become lost very easily. There is a large body of research that attempts to address these issues and this area is not the focus of the portal component. The portal component is a simple mechanism to provide users with the ability to move between locations in a 3D space. In essence it is a simple gateway from one location to another. Other systems for providing such gateways have been developed including the various forms of WIM (i.e. World in Miniature) [Pausch 95], [Stoakey 95], [Elvins 97], which provide the users with miniature representations of other locations into which they can move.

The portal component is a very simple variant of these systems and provides the user with a gateway from one location to another. The physical structure of this gateway provides the user with a sense of the content at the new location (as shown below). Developed primarily to provide a functional tool to enable users to jump between workspaces (i.e. this component was constructed more through the necessity of arranging the workspaces than any particular research objective) and as such this component is simple and targeted at this specific function (unlike the more complex WIM systems which also enable user navigation in the existing space).

Essentially the portal described here represents a component which can be used to provide this simple multi-workspace navigation system (i.e. a doorway between workspaces which gives a sense of the content in the target workspace).


**Figure 6.53: The Portal Component**

As the diagram above shows a portal is simply a ring on the inside of which is a miniature version of the world at the other end of that ring (currently this miniature

version is manually created by the developer, however there is the potential to automate its construction in future versions). By clicking the ring or miniature world the user is moved to the other location (i.e. in the diagram the user would be moved to Workspace A). Although incredibly simple this representation is effective in its task (note that the physical ring can be changed to one of a range of variants with the initial versions implemented as ring, doorway and globe).

These portals can be placed anywhere in a 3D space to enable users to jump to varied locations. In addition to the possibilities of placing such components in fixed locations, they can also be attached to the user him/herself (through the toolbelt component described below) and as such can provide the user with a simple method of moving between common workspaces or locations (whether those locations are simply other viewpoints within this world or are pages in the wider Internet).

## The Toolbelt Component

The concept of a user carrying tools with him/her is not new (especially as it is something all people undertake in the real world on an every day basis, whether it be using pockets to hold keys or carrying large items). However implementing this principle in a 3D user interface is an area that is still under development. Starting with the implementation of the "virtual hand" [Fisher 89], to provide the user with a presence in a VR system. This was quickly extended into a hand with a tool. Perhaps the most widely used implementations of such systems occurred in the games industry [Wolfenstein 92], where the "virtual hand holding a weapon" has become the standard interface for most 3D games today.

Most recently [Pierce 99], [Robertson 00] discuss the concept of "toolspaces" as regions around a users location in space into which tools and files can be stored. By combining these "toolspaces" with a "glance" technique they describe a system for providing small storage regions that move with the user and can be viewed via a simple "glance" action. Hence enabling the users view to remain uncluttered, yet when required, these toolspaces can be viewed through a simple glance action. Essentially these toolspaces are described as simple empty regions.

> ' We believe that there are several potential avenues for future work. One is to explore other layouts and constraints for placing objects in toolspaces, especially if large numbers of objects need to be stored in a small number of toolspaces. We believe that different tasks will require different layouts and constraints, and determining the correct design choices will require careful experimentation' [Pierce 99].

It is this organization and arrangement of items that the tool-belt and its sub components (i.e. the Pouch and the Swiss Army Knife) address. Essentially the key concepts for the components developed in this section are to extend the concept of "move with user" items to provide useful mechanisms for arranging and presenting both the tools and other items that a user carries with him/her through space.

The toolbelt is essentially a simple grouping component, designed to enable the user to attach items of varying types to his/her "move with users" set. Much like a real world belt, the toolbelt enables users to attach (or hang) individual items, groups, containers as well as tools onto the toolbelt and have them move with the user through space.

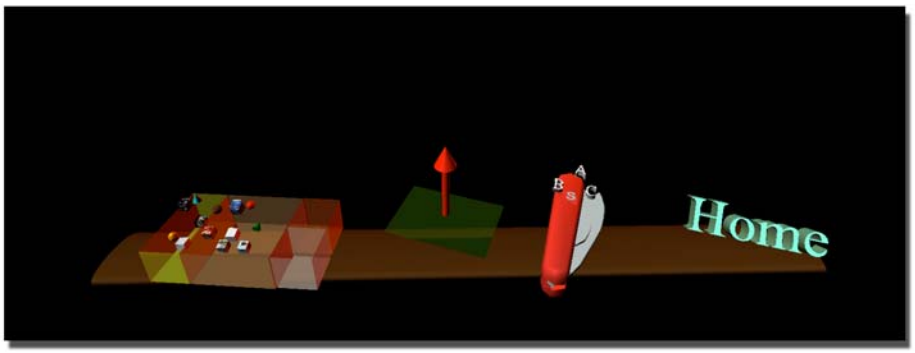



**Figure 6.54: Screenshot of a toolbelt.**

This system also ties in well with [Mine 97]'s concept of keeping a users tools in close proximity to his/her body (e.g. in the users hands).

As the previous diagram shows the toolbelt itself is simply the brown flattened tube that is visible below the other items. each of those items is attached as a child of the toolbelt and thus when the user moves, the whole structure moves with the user (much like an item of clothing moves with its wearer in the real world). This toolbelt can be held either on-screen (i.e. in the users view as shown above) or off-screen (i.e. in an adjacent space much like those described in toolspaces). As such the user can choose whether to have the toolbelt always visible or only visible when requested or "glanced at".

Another key design issue for the toolbelt is the decision to separate tools and items. Although this separation can sometimes be challenging, in general it is a simple process for users or developers to determine which items are tools (eg. applications that can act on other elements) and which are simply content (i.e. files of data, images etc). By providing independent storage locations for the two different types of items the user can more easily determine where to look when seeking a tool and where to look when seeking or storing an item.

Each sub-item that is attached to a toolbelt can be of varied complexity, there are no particular restrictions and as the whole system is designed using reusable components it is entirely possible for a sub element to be a large and complex system or component (eg. a circulatory system). The previous diagram shows a toolbelt with a Pouch (container for storing items) on the left, then an orientation aid, then a "Swiss Army Knife" (i.e. this is a storage component for grouping and displaying tools or applications) and finally a link to "Home" which will transport the user back to a known location. Using specialized grouping components, such as the pouch and Swiss army knife enables the basic toolbelt to achieve the desired tool and content separation through the use of independent and separate storage components. By both being children of the toolbelt these groups can be retained near each other and readily available to the user at all times.

***The Implementation***

The toolbelt is designed to provide a generic reusable component for the task of grouping items and linking them to a user as he/she moves through the 3D environment. Constructed using the builder tool, the toolbelt appears to be a relatively simple component from the developers perspective, as all he/she needs to do is add children to it and they automatically attach to the users movement. However the implementation is a little more complex. The toolbelt is constructed as a simple parent component with a primary purpose to provide a mechanism for its child items to be attached to the users location and hence move with the user through space. The physical structure of the toolbelt is a simple cylinder flattened geometrically to provide a platform onto which other "tools" can be added. This physical "belt" item is actually optional and can be removed, thus making the toolbelt simply a grouping structure with no physical presence.

The key implementation issue related to the toolbelt is its capability to remain attached to the users view. This attachment is achieved through the use of a proximity sensor in the toolbelt itself. The proximity sensor is sensitive to user movement within its range and thus every time the user moves or rotates the proximity sensor detects the movement.

When a movement is detected the proximity sensor gets the users new position and orientation. Using this information the position and orientation of the toolbelt can be adjusted to match its location to that of the user (i.e. by matching the toolbelt this also matches all of its child objects). This simple system provides an efficient mechanism for implementing the “move with user” functionality required.

***Summary***

The toolbelt is simply a component that moves with the user through the 3D space. By linking other items to this “toolbelt” component a developer can construct a set of “move with user” items. Essentially this component provides the interface developer with a mechanism for implementing the “move with user” functionality and from this base a set of more complex components can be constructed to help arrange and present these “move with user” tools more effectively. Used in combination with these “arranging” tools the toolbelt becomes a powerful resource for the user of a 3D environment as it enables them to carry tools and items of varied types with them as they navigate through worlds.

## The Pouch Component

The pouch is designed to be a component for storing items in the short term in a semi structured manner. With the introduction of the mobile user concept there is a need to enable the user to carry items with him/her as he/she navigates through 3D space. Existing methods such as “toolspaces” offer the user a simple 3D volume to place items into, however this unstructured approach is less than ideal for arranging information (especially large sets). In many ways an unstructured space of this type suffers the same problems as those regularly seen on the desktops of many GUI users. This is clearly reflected in the common cluttered desktop of today’s GUI users.

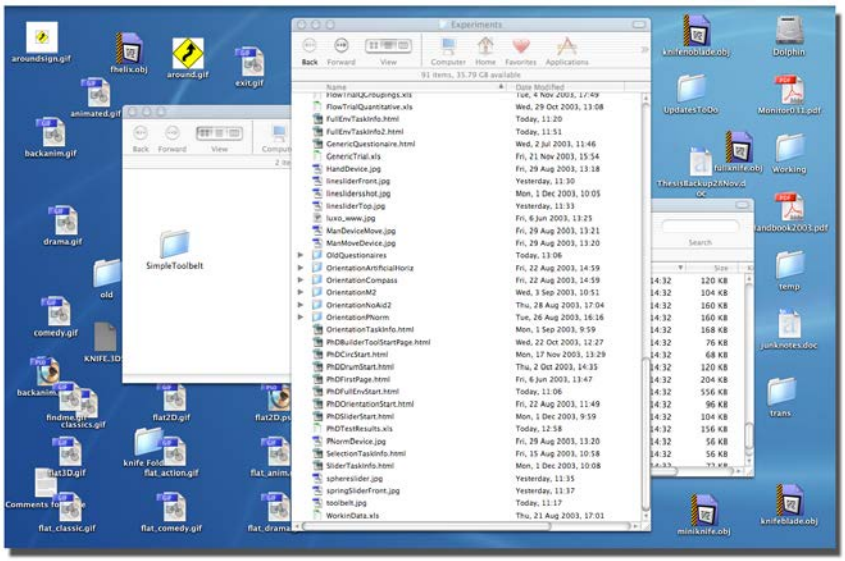


**Figure 6.55: The common (cluttered) GUI desktop.**

As the diagram above demonstrates, users have a tendency to grab items when they see them and rather than taking the time to find the best location they simply place them in a

readily available space (i.e. the GUI desktop) for detailed placement later (something which rarely occurs).

Although it is appealing to simply indicate that users should place items correctly, the reality is that they do not and hence the 3D mobile user (like his 2D GUI equivalent) will also require a place to store items of this exact type. Filling up the users view in 3D will simply block his/her view and hence is unsuited. Unstructured spaces such as those described in [Pierce 99] will quickly fill (like the desktop) making them unusable beyond small sets of data. For this purpose the Pouch component described below is designed.

### *Description & Implementation*

The pouch is a simple container based item. Designed to offer the user a space (or set of spaces) into which items can be placed. Each sub-container is capable of holding a set of items and interactively displaying those items to the user when required.

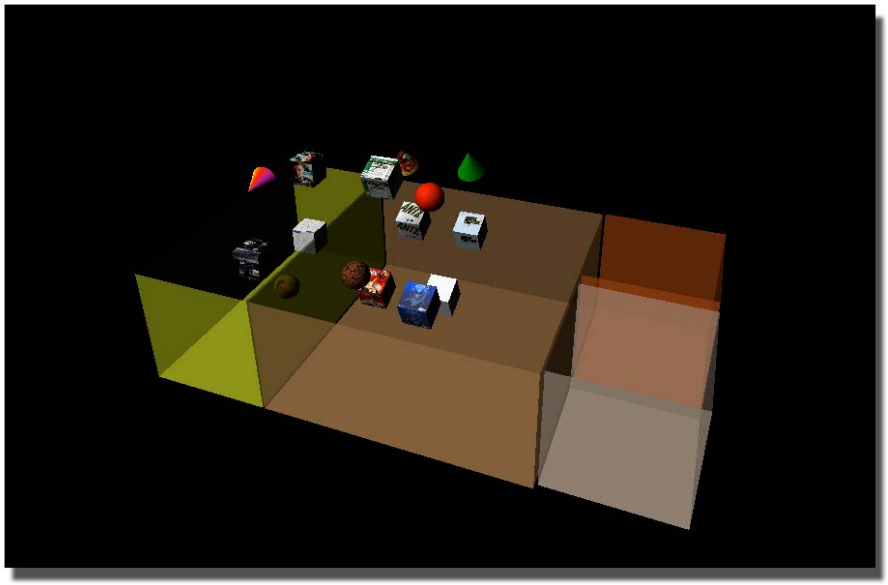


**Figure 6.56: The Pouch Component**

The pouch component consists of several key elements. Firstly it is made up of a series of boxes (i.e. the yellow box on the left above is an example of a base box or compartment). Each box represents a single storage space into which items can be placed an retrieved. These boxes can be resized, arranged and personalized to suit a users preferences (eg. have unique colours, sizes and even shapes). Utilizing multiple pouches can also offer an additional organizational capability.

The key objective with this multi box based component is to provide the user with a simple mechanism for storing items in a semi structured form without going to the lengths of saving it into final locations. In essence it is a short term storage location for holding items so that they will be readily accessible. This type of storage enables the user to simply grab a file (eg. from the web etc) and place it into a box (or compartment) of the pouch.

In addition to a basic storage location there is also a need to enable users to keep this data organized. At least to the point where they can relocate these items at a later date. Taking basic design principles for organizing data (eg. unique compartments to arrange items based on users personal grouping style) the pouch provides several compartments of differing shapes and sizes and allows the user to create a layout of compartments based on their personal style. The fact that the pouch component can be attached to the toolbelt and hence move with the user makes it always on hand for storing or retrieving items (i.e. much like a handbag/pouch/sports bag it provides a structured short term space that's always on hand for dropping things into or getting them out).

The actions involved in adding or retrieving items from the pouch are relatively simple and involve the user dragging items into a compartment of the pouch to add them or simply clicking on the item in the pouch to retrieve it. In order to present the available items to the user the pouch uses an innovative cycling system. For the retrieval task the user would move the mouse over the compartment causing the set of items to cycle in a loop from the bottom of the compartment through to the top (as shown on left), thus enabling large sets to be able to be stored and retrieved. This cycling system is similar in implementation to the ring component that is used in the circulatory system and as such it is possible to use a "ring component" instead of a simple cycling set and thus the user would be able to click the ring to move inside it and have the data in that set flow past as described in the circulatory system component earlier in the thesis. The basic ring based layout of the items in the set enables the user to see the entire set, although some will be in the base of the compartment and others on the top. by providing the simple cycling system the user can access all items in a smooth manner. With the ability to use a "circulatory system ring" in place of the basic cycling set the developer is able to create a compartment which can be used to present large sets of data.

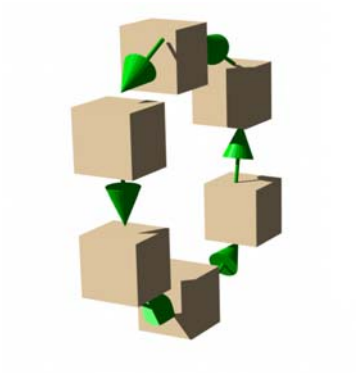


**Figure 6.57: Cycling of data in a pouch compartment**

***The Implementation***

The pouch component, although simple in concept consists of several relatively complex elements. In implementing its overall structure the pouch is made up of a series of sub components which come together to form the final component. This component based approach is helpful both from an implementation perspective and from a reuse perspective (i.e. as the user/developer can take the compartment components and rearrange them based on personal preference rather than having a fixed layout/scheme).

At the simplest level are the "base items" that the pouch stores, these are simply represented by boxes with unique texture maps and actions when clicked (i.e. enlarge when isOver and run action when clicked). The next level up is the "cycling set", this is simply a set of base items that are grouped into a set and laid out in a circular pattern. Abstracting one level further is the compartment component, this component has the physical structure of an open topped box (two versions are implemented, the cube and the

cylinder), by utilizing the ability to scale and change the surface attributes these generic forms of box can be altered to produce a range of differing box shapes, sizes and appearances. In addition to their own structure the compartment also contains a child "cycling set", consisting of the items in this compartment. These boxes are also sensitive to the users moving the pointer over them and when that event is detected they spin their "cycling set" about its circular axis, thus cycling the items from the base of the box to the top.

The other key piece of functionality provided by the compartment component is its ability to detect when items are dragged into its space. Implemented using a simple proximity sensor (which is attached to the physical structure of the box) the compartment is able to detect when an item is dragged within range the compartment and then add that item as a child of its cycling set (i.e. resize then place it in the set).

Finally the parent item is the pouch itself which is simply a grouping item that links the compartments together and enables the set to be manipulated as a whole and thus added to items like the toolbelt as a single piece.

***Summary***

The pouch provides a simple to use, yet powerful and easily personalized mechanism for providing users of a 3D interface with a form of storing files for quick and easy retrieval. This work contributes to the area of 3D interfaces through its identification of the need for "move with user" components that empower users to carry the tools and items they need with them through 3D space. In particular the pouch provides a new component for the storage and retrieval of items from a readily available storage mechanism. By providing a new and innovative component to handle this task the pouch extends such concepts as the toolspace [Pierce 99] by taking the empty spaces and providing a simple, personalizable storage system in the form of a reusable set of components which can be applied in a range of 3D user interface applications.

## The Swiss Army Knife

Essentially similar to the pouch this component provides a grouping and presentation tool for the presentation of applications and tools. Functionally the pouch and knife are very similar in their implementation. Both systems provide a grouping mechanism for collecting items and both provide mechanisms for adding and retrieving items from those sets. The key differences are in the appearances of the groups and the fact that the knife offers less sub-categorizing capability.

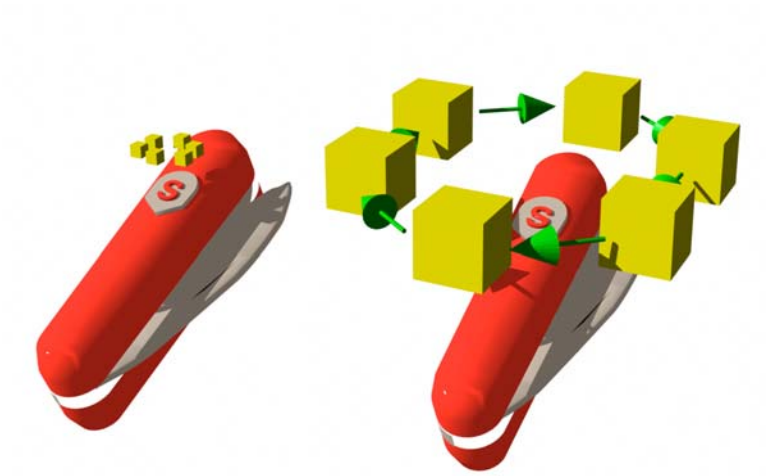


**Figure 6.58: The Swiss Army Knife component (left is unselected, right is selected)**

As the previous diagram shows the knife consists of a physical model of a Swiss Army knife. This visual representation was chosen as it represents a tool that holds many other tools and as such would be quickly recognized as a place to look when seeking a tool.

The physical knife object is simply a base onto which the set of available tools are added. Unlike the pouch the knife only offers one level of sub-grouping and hence all tools must fit into rings at the level below the knife itself. The set of tools enlarges in place and spins as shown in the previous diagram when the knife structure is “moved over” by the users mouse/target. This mechanism enables the user to view the available tools and make a selection. The size of individual tools can be set by the developer as can the number of rings of tool options (i.e. in some cases tools may be grouped into categories each represented by individual rings). This system of displaying the tools only when the user moves over the knife enable the set of tools to always be readily available yet only become large (and hence take up larger part of view) when requested.

This component is implemented using similar techniques to those described in the pouch (see previous section for more details). The knife component essentially provides the user with a single location from which to locate and access his/her tools. When this component is added to a toolbelt (and thus moves with the user) it enables the user to carry his/her tools with him through the 3D environment and thus when needed they are readily at hand.

## Summary of Set of Full Environment Components

With the potential of a future filled with networked systems where the local machine will no longer represent the user (this is already becoming more and more the reality today), in fact the user is more likely to be moving between multiple systems. The ability for that user to carry his/her tools and items with him is a critical piece of implementing effective 3D interfaces. In addition the future offers the potential of systems such as augmented reality (where the 3D interface is overlayed over the users view of the real world). In such systems the importance of developing effective and functional “move with user” tools and interactive systems is critical. This research has identified and developed a range of new components designed specifically to target the common tasks that a user

requires from these “move with user” tools, including the ability to store and retrieve items through the pouch and the ability to carry and use tools through the Swiss army knife component all tied together and attached to the user through the toolbelt component.

## Discussion

Each of the components described above is intended to provide specific functionality, however one of the most difficult challenges in creating these components is involved in indicating the function of the component to the user. [Snibbe 92] introduced the concept of “Self Disclosure” as a widget whose geometry indicates its behaviour. This concept is used for all of the components in this section in particular the pouch (implemented to look like a set of cardboard boxes or containers i.e. items into which other items can be placed) and the Swiss army knife (implemented as a geometric Swiss army knife i.e. to indicate that it is a tool that contains other tools) utilize this technique. Another key design issue created by these particular components is the decision to separate tools and content through the use of differing grouping components. Used effectively such a system can aid users in understanding where to look when seeking particular items, however there is also the potential for confusion (i.e. the user assumes the Swiss army knife is a cutting tool and looks in pouch for the desired tool). This issue can be easily addressed through experience, however it identifies the key challenge of using “Self Disclosure” as much as possible without assuming it as a guaranteed mechanism to communicate a components purpose.

These components provide a powerful mechanism for attaching items to a users location as he/she moves through space, however they still face implementation challenges such as whether they should be permanently present in the users view or not. By placing them out of view as proposed by [Pierce 99] they do not take up valuable viewing space, yet by being off screen the user may not utilize their potential. Providing the user with flexibility (i.e. choose between in view and out of view) appears the best option and as such is the version implemented here.

### *6.4.9 Component Set Summary*

The previous sections describe a range of different components each designed to handle a common real world task. Each of these components was designed and constructed to produce general purpose reusable components, which in combination with the builder tool described earlier in this research provide an overall development platform from which a range of interactive 3D interfaces can be constructed. The diagram below gives a graphical representation of the full development platform and the components from which developers can construct new systems.

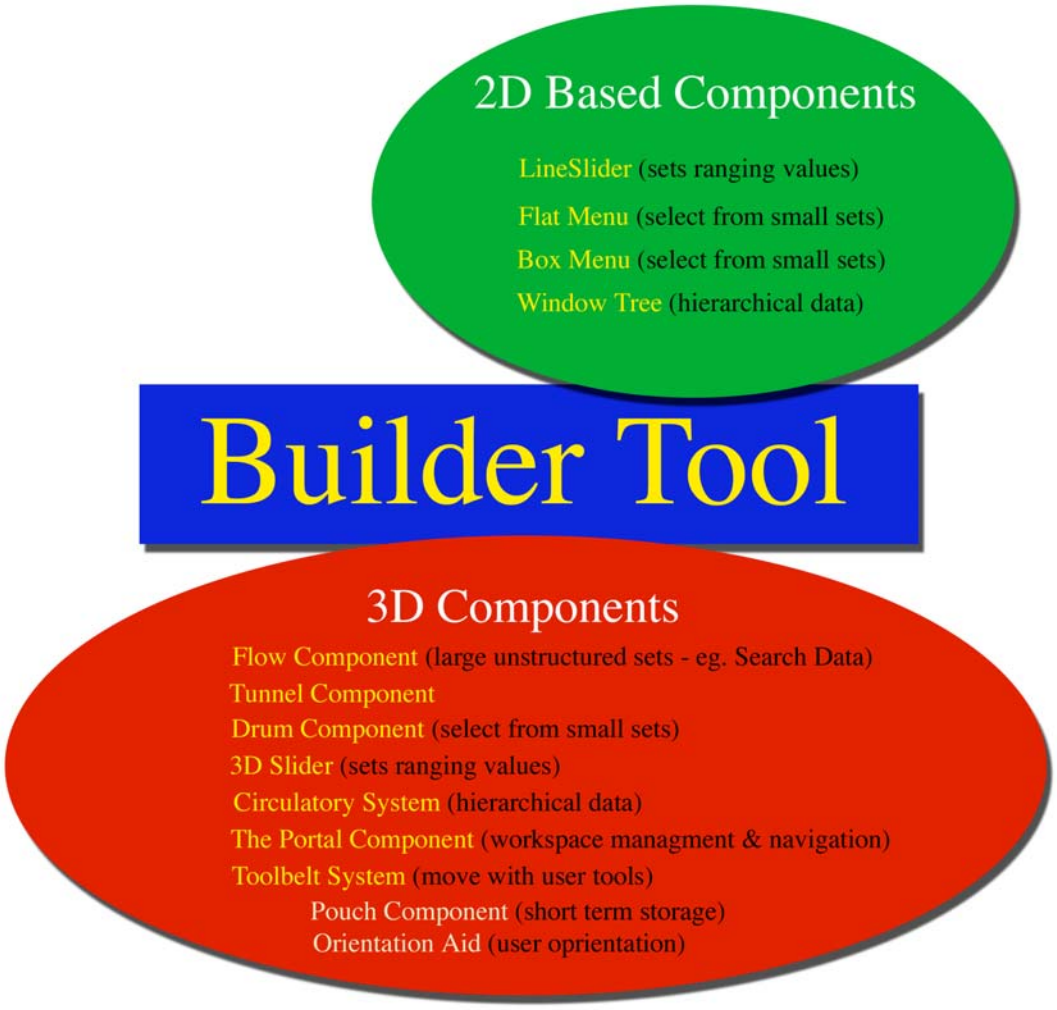



**Figure 6.59: The full component set.**

The “2D Based Components” as shown above represent the set of components that are based on existing two dimensional techniques and have been altered slightly to enable their implementation in a 3D interface toolkit. Of the remaining components shown above all of those described in the red “3D Components” section represent new conceptual interfaces developed as part of this research. These components contribute to

the field of 3D interface design by describing new and innovative mechanisms to handle such common interaction tasks as:

- Selection from search data (i.e. large unstructured data sets using the flow & tunnel components)
- Selections from hierarchical data (i.e. structured data sets, such as library data using the circulatory system)
- Setting ranging values (i.e. set values in fixed ranges using the 3D slider)
- Selections from small sets (i.e. using the drum)
- Organization and navigation between 3D workspaces (using the portal component)
- Tool-belt to enable the user to carry tools with him (using the tool-belt, pouch and orientation aid)

These components when put together create a broad set of tools from which many 3D interfaces can be easily created. In combination with the builder tool which enables the user to rapidly bring these components together into a complete interface, this overall development platform provides a strong base from which new and innovative 3D interfaces can be created.

# *Chapter 7: Evaluating 3D SPACE - Testing On Users*

## 7.1 Experiment Design

This section describes the experimental testing approach that was used to determine the effectiveness of the 3D components as described in the earlier sections. To achieve this each 3D component was tested with real users, undertaking practical real world tasks. These experiments all follow a basic outline involving the user completing a task with the relevant 3D component and also completing that same task with a number of other interface versions (i.e. some 2D systems, others simply variants of the primary 3D component). During these interactive tasks, performance and related measures are recorded thus providing a set of quantitative results. From these results it is possible to determine the relative effectiveness of the 3D components. In addition to the results mentioned above, at the end of each task the users are required to complete a questionnaire, thus providing an additional set of qualitative result data. For a more thorough description of the testing process see the *Appendix A: User Testing Setup* section at the end of this thesis.

The overall testing involved fourteen trial users[31], all over the age of 18 years and all volunteers. Each specific trial was undertaken by between 8 and 12 users (with exceptions such as the builder tool (i.e. the case study trial) noted in the results below). The trials involve no special training or experience with the 3D systems prior to testing, in fact having agreed to take part each participant simply reads the *Explanatory Statement for Research Participants* (see *Appendix B* for this document), signs the consent form and begins the testing process.

The complete testing set consists of a group of independent experiments, each running for a period of approximately 30 minutes and each focussing on a particular component or task. Each trial is limited to approximately 30 minutes to ensure that the participants remain fresh and that their responses do not reflect any fatigue. Each of these 30 minute trials/experiments follows the basic pattern as outlined in the following section.

The experiments are all designed to isolate one feature of an interface (eg. take the same task with the same items only presented in a different manner). To achieve this all other factors are kept consistent across the trials, including the hardware, device set-up and software systems used. By maintaining this consistent system and having the trial interface as the only varying feature conclusions can be reached regarding the relative performances of the systems and the effectiveness of the various forms of presenting data to the user.

---

[31] Note that as this is a voluntary process not every user completed every task/trial.

## 7.2 Experimental Results

The user testing process records two forms of result data. The first is the interactively recorded set of performance figures. These results are recorded while the user interacts with the 3D system and include quantitative values such as time taken to complete tasks, number of incorrect clicks/actions, number of correct clicks and so on. The second set of recorded data is the set of questionnaire responses. This qualitative set of results is recorded when the user submits his/her answers to the questionnaire form (which is presented following each interactive system).

Using this combination provides each interface system with a set of both quantitative and qualitative results which can in turn be directly compared to the other systems (which are recording the same sets of data) used in the trial. These results enable comparisons to be made in a number of ways including, single user system to system comparisons (performance comparisons between the results achieved by single users when using differing interfaces) and also multiple user system by system comparisons (performance comparisons of results of all users for one specific system versus another).

In addition to this basic initial set of result data, this project was interested in observing the effect that experience with these 3D interfaces would have on performance and usability. To achieve this a subset of the initial user group repeated the trials (after a reasonable period away), thus providing another set of comparison data enabling the comparison of “single user first run” versus “single user second run” results. The only difference in the second set of figures is that caused by the users increased level of experience with the 3D interfaces and this provides some interesting results.

Each experiment recorded differing data, however there are some values that were recorded across all trials (eg. Question: “How Effective was this Interface for this task?”). Essentially the trials all followed the basic layout as described below:

- **Initial “Welcome Page” displayed to the User -**

  Information Supplied:

  Description of Questionnaires (i.e. 1-20 scale ratings).

  Information Obtained:

  (Quall) The users experience with Computers (1-20)

  (Qual) The users experience with 3D software (1-20).

  Notes:

  This page remains consistent across all of the trials.

- **Task Specific Information Page displayed to the User -**

  Information Supplied:

  Description of the task the user must complete (e.g. locate and click this item in a set).

  Information Obtained:

  n/a

  Notes:

  This page varies based on the task and interface to be used.

- **Complete Task with Different Systems** - For each system to be tested do the following:

  **Display Interactive 3D Interface to the User -**

  Information Supplied:

  Presentation of the data in its 3D interactive form.

  Information Obtained:

  Interactively recorded quantitative performance results (e.g. time taken, errors etc).

  Notes:

  The interactively recorded values vary from experiment to experiment, however within any one experiment the data recorded is the same for all interfaces. Thus enabling comparison of the system results.

  **Display Questionnaire to the User -**

  Information Supplied:

  n/a (questionnaire is entirely questions)

  Information Obtained:

  Qualitative responses to questions in the questionnaire.

  Notes:

  The questions posed and hence the recorded values vary from experiment to experiment, however within any one experiment the data recorded is the same for all interfaces. Thus enabling comparison of the system results..

- **Task Wrap-up Questionnaire Page displayed to the User -**

  Information Supplied:

  n/a (questionnaire is entirely questions)

  Information Obtained:

  Qualitative responses to questions in the questionnaire.

  Notes:

  The questions posed in this section are generally broader in scale. They often ask the users to select their favored interface from those that were trialed or to identify the suitability of 3D to the task

- **Final "Thankyou Page" displayed to User -**

The following sections present each experiment in greater detail, including the specific details of the tasks and values recorded in each trial. In addition the results from those trials are presented in the form of charts showing the average result and the spread of results (i.e. the 95% confidence interval) relative to that average (eg. see Figure 7.1)

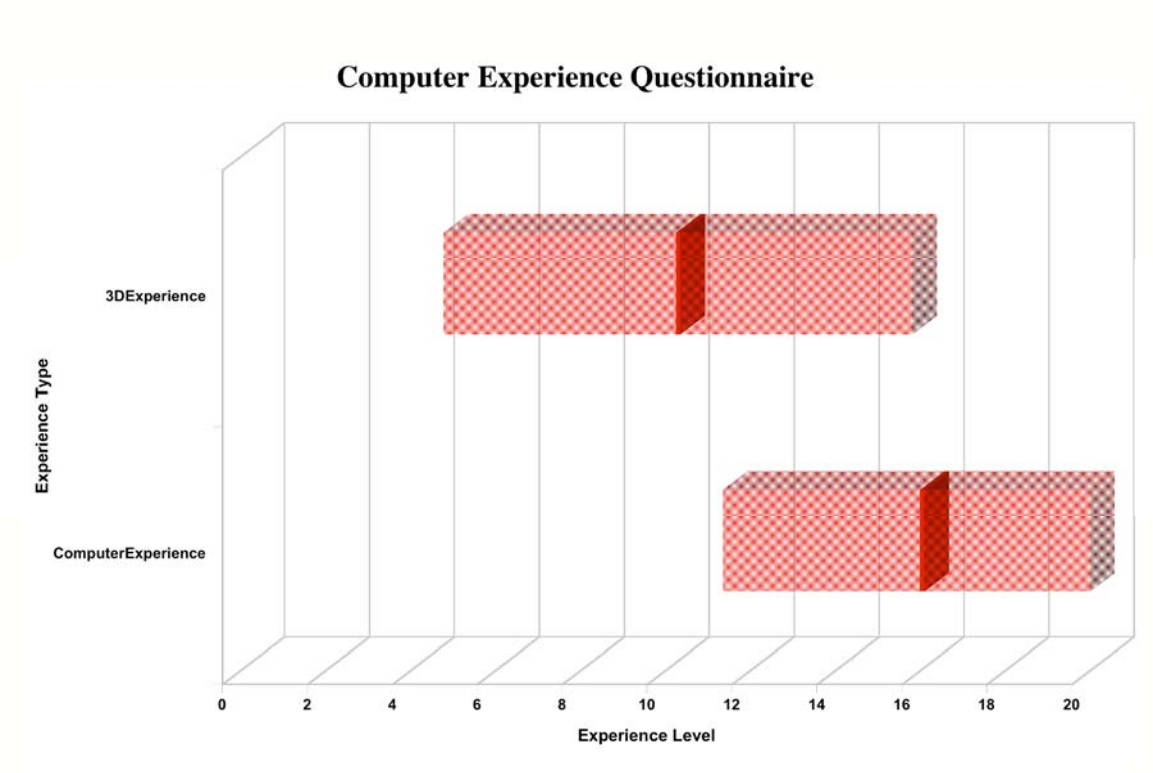



**Figure 7.1: Example Result Chart**

The chart above, which shows the user responses to the initial questions, “How much exposure have you had to Computer Systems?” and “How much exposure have you had to 3D computer software (eg. games, animation/modelling applications)?“, demonstrates this display method.

On this chart you can see the “95% confidence interval” shown in the form of the partially transparent red bar. Within this confidence range is the solid red bar representing the average or mean of the users test results. The intent of these charts is to graphically display the statistical results for each trial, demonstrating both the average result and the confidence interval associated with that result. Statistically this 95% confidence range is calculated by taking the range of values 1.96*SE (i.e. SE - Standard-Error) either side of the mean (for more information on this type of statistical analysis see [Streiner 96]).

From a statistical perspective this would be interpreted as meaning that we are 95% confident that the average value of the result in the population of trial participants is somewhere within this interval. This statistical/charting method will be used to display results for all of the trial systems.

At the beginning of the trials each user is asked to identify their existing level of computer and 3D experience, ranging from 0 (i.e. never used) to 20 (i.e. regular user). The results from these questions provide a valuable insight into the user/survey group. The chart above (i.e. Figure 7.1) displays the results from these computer experience related questions. This chart shows that the user group is relatively experienced with computer systems (i.e. mean value of 16) yet less experienced with 3D systems (i.e. mean value of 10). The 95% confidence intervals also provide interesting information because

they identify that the groups range of 3D experience varies more than its computer experience. Overall this indicates that the trial participants largely consist of relatively experienced computer users with a lower level (yet more varied between participants) of 3D experience. The following sections outline the specific trial systems and the results achieved by users when completing tasks with the various 3D interfaces proposed in this research.

## *7.2.1 The 3D Components*

The full set of 3D components provides a broad range of items, each intended to provide an effective mechanism for 3D interaction in a particular task. The following sections outline the tasks used to assess each component in terms of effectiveness and usability.

### Assessing the Flow Component

The flow trial is designed to test a range of different methods for selecting specific items from within a large set of unstructured options (eg. search results). For this trial the task is based on the real-world task of locating a specific result (which occurs multiple times) from within a broad set of web page based search results.

As described in the earlier section, the "Flow component" essentially presents the data as a flowing stream of items, somewhat like a river of items flowing past the user in 3D space. To assess the relative effectiveness of this "Flow component" it is compared to a set of other interfaces for presenting the same kind of large, unstructured data.

#### *The Task*

The user is asked to locate and select (by clicking) all occurrences of a particular web page from within a large unstructured set of web pages.

**Figure 7.2 Example Item**

This set is represented in the form of a collection of three dimensional cubes (as shown on right), each identical to the others with the exception of its texture map. Each such cube represents one web page link (which if clicked would take the user to the specified web page) from the set. For the purposes of the trial when clicked these objects do not have associated web links instead they simply register either as correct hits or incorrect hits (depending on whether they are the desired target or not). Using screen captures from a variety (50 unique web page texture maps) of web pages this group of items, when randomly mixed, provides the "large unstructured data" set for this selection task.

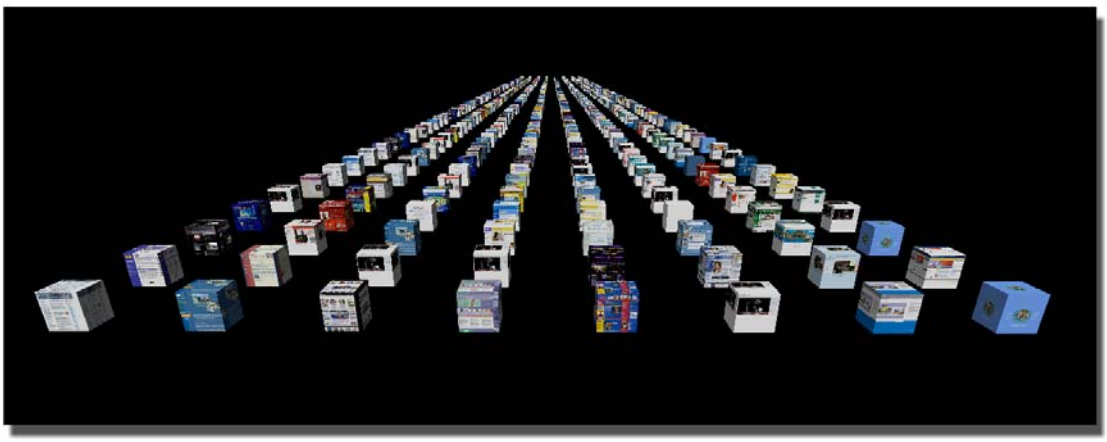


**Figure 7.3: Example of Set (shown with "In to Out" Flow arrangement)**

The set itself contains 640 items into which 72 target items have been randomly inserted. The target web page as used in the trials is shown below on the left (Figure 7.4). This page was randomly selected from the available range of 50 choices.


**Figure 7.4 The target web page**

The final set, including the target items, is then presented using a range of differing methods to enable comparison of the users performance with each. For all tests the set of boxes flows at the same speed and has the same density, arrangement and placement of target items. The only difference is the type of presentation of the set.

The set is presented using seven different interfaces, ranging from essentially two dimensional scrolling lists to highly interactive 3D systems. The user completes the selection task with each system to provide a set of comparative performance and opinion data.

### *The Seven Trial Systems*

The trial involves the participant using each of seven different interfaces to undertake the selection task (i.e. select the target web pages). These interfaces fall into two key categories, firstly the fundamentally 3D systems (IntoOut and variants) which use the depth of the screen to full effect in presentation of the data set. The second set of interfaces are the fundamentally 2D systems (ToptoBase and LefttoRight) which essentially present the set square on to the users view much like a 2D scrolling list, only constantly flowing. The specific system tested in the trials include:

- **Top to Base (Sparse & Dense Variants):**

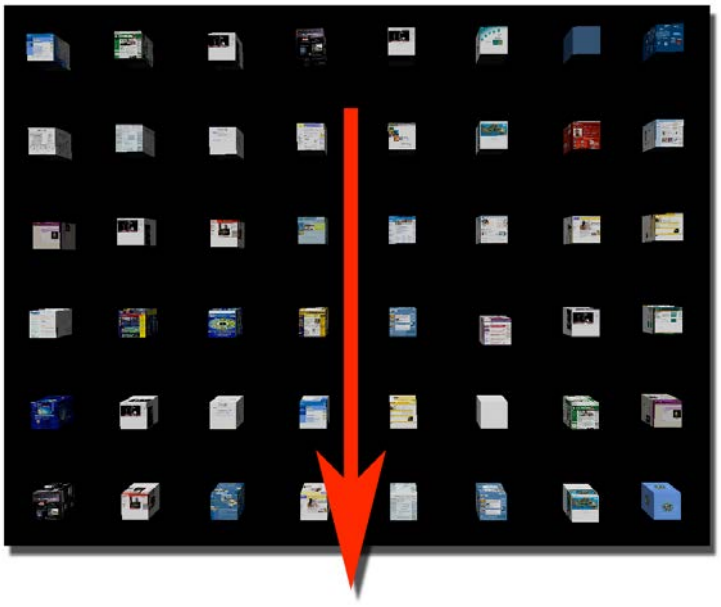


**Figure 7.5: Top to Base (Sparse) Flow**

As the diagram shows the items are aligned across the top row of the space and move downwards on a plane that is parallel to the cameras plane, thus giving the effect of items staying an equal distance away from the user and scrolling from the top of the view down and out of view at the bottom. Given that the user is looking squarely onto one face of the cubes this form of presentation is very similar to a 2D scrolling list (except that it is moving). There are two variants of this interface, the “Sparse” version where rows 8 items wide are fitted across the users view (as in diagram above, giving 6 rows in view at one time). The second variant is the “Dense” version where the rows are 8 items wide but also 8 items high (giving a larger set (64 items) set of smaller items).

- **Left to Right (Sparse & Dense Variants):**

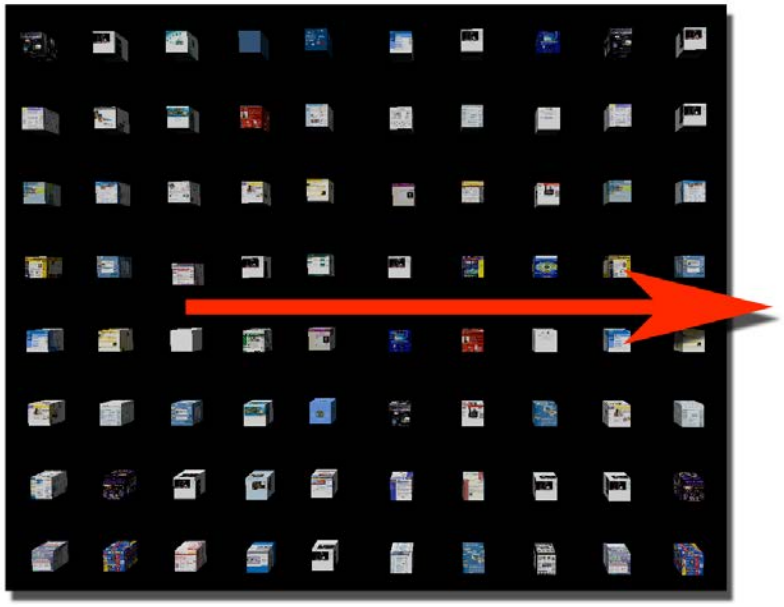


**Figure 7.6: Left to Right (Dense) Flow**

For this interface the items are aligned along the left hand edge/column of the space and move to the right on a plane that is parallel to the cameras plane, thus giving the effect of items staying an equal distance away from the user and scrolling from the left of the view across to the right of the view and out of view at the right hand side. As with the ToptoBase system there are two variants implemented, “Sparse” (fewer larger items) and “Dense” (more smaller items).

- **In to Out (Flat):**

**Figure 7.7: In to Out (Flat) Flow**

The first of the fundamentally 3D interfaces, this system has items that are aligned along a plane that extends like a ground plane below the user and out towards the front of the screen. A series of rows (8 items wide) steadily extending, row by row deeper into the screen. These rows move from the depth of the screen out towards the front (i.e. near plane) of the screen. Thus giving the effect of items moving from the depths of the screen to the shallow/near part of the view and then through and under the user.

- **In to Out (Left):**

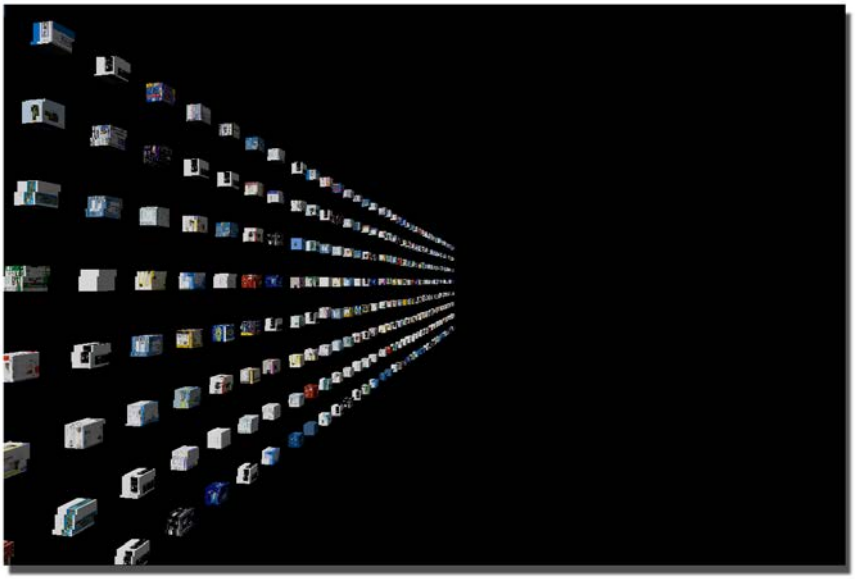


**Figure 7.8: In to Out - Left (8 High) Flow**

In this 3D depth oriented interface the items are aligned along a plane that extends like a wall on the side of the user and back into the depth of the screen. A series of columns (8 items high) steadily extending, column by column out from the depths of the screen. These columns move from the depth of the screen out towards the front (i.e. near plane) of the screen and also shift from the users left to the users right during the process. Thus giving

the effect of items moving from the depths of the screen while also shifting subtly across the users view from left to right.

- **In to Out (Spinning):**

**Figure 7.9: In to Out (Spinning) Flow**

In this, the most complex of the 3D interfaces, the items are aligned along a plane that extends back into the depth of the screen. A series of rows (8 items wide) steadily extending, row by row forward from the back of the screen. Initially appearing much like the "IntoOut (Flat)" interface. However unlike the other "In to Out" methods, this system not only moves in the depth plane (i.e. from depth to near) but also rotates the set about its Z axis (i.e. spinning in a clockwise direction from the users perspective). It also moves the set across the users focal point (i.e. going from flowing below and under the user to flowing above and over the user).

### *The Trial Itself*

The interactive trial involves each user being exposed to the selection of seven "flows" for a period of exactly 60 seconds each (in which time 320 of the total 640 items will flow past the users position). In that time the user will select as many target items as possible. While the user undertakes this task the following data is interactively recorded:

- Number of Correct Hits
- Number of Repeat Correct Hits (i.e. did user double click it)
- Number of Incorrect Hits

Throughout this trial the user is not provided with "flow controls", despite the fact that these are a feature of the "flow component". This enables each trial system to be compared given identical flow rates and isolates the presentation of the flow as the only varying factor.

The trial involves a relatively high density of items which move past the user relatively quickly. It is intended to be a difficult task for the users to select every target item as this provides more useful comparisons (i.e. if the density and speed were slowed users would correctly identify and select all targets in all systems thus providing little useful comparative information).

To measure the success of each of the trial systems essentially involves comparing how the users perform the selection task using the different types of flow interface. In addition to this quantitative data the trials also incorporate a series of questions, which are asked following the user completing the task with each trial system. This combination of data enables the results to be assessed to determine if the flow is effective and in which arrangements it is most/least effective.

## Results for Flow Component

This section provides the result data from the flow trials, including both the quantitative and qualitative results. This trial was undertaken by eleven participants, with each participant completing the selection task with all seven trial systems. The systems were presented in the same order to all participants, beginning with LefttoRight (Sparse) and ending with IntoOut (Spinning), see Figure 7.10 for the specific order.

### *Quantitative Performance Results*

The quantitative results for the flow trial represent the users performance data, recorded while the user completed the selection task. The results were as follows:

- **Number of Correct Items Selected**: shown below using the 95% confidence interval (partially transparent range) and the mean (solid red line).

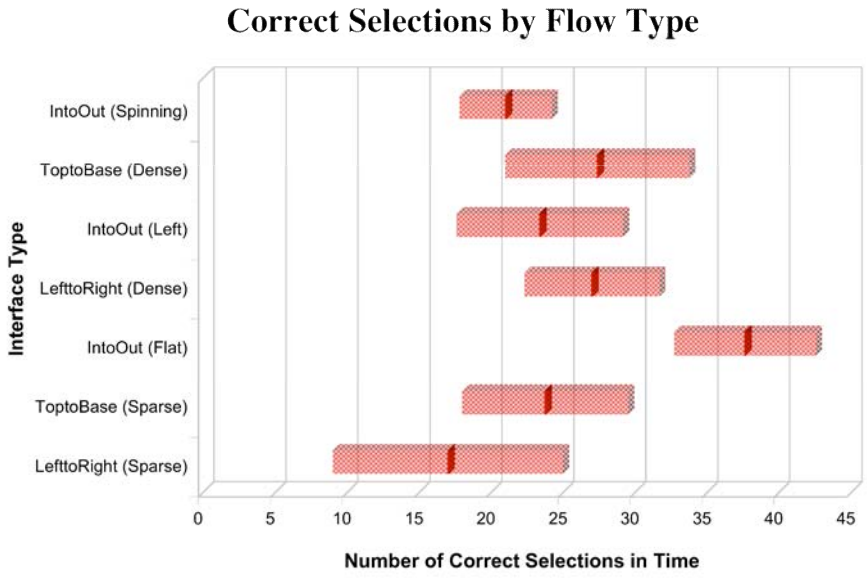



**Figure 7.10: Correct Selections by Flow Type**

Figure 7.10 above demonstrates the measure to which users were able to effectively perform the selection task. With only one system showing any significant advantage when compared to the others. These results demonstrate the fact that the highly three dimensional "IntoOut (Flat)" method was the most effective. This method was clearly more effective than the others with the next closest being the essentially two dimensional (i.e. very similar to a 2D scrolling list) "ToptoBase (Dense)" and "LefttoRight (Dense)". Interestingly not all of the three dimensional methods proved effective with the "IntoOut (Spinning)" and "IntoOut (Left)" showing relatively poor performances.

- **Number of Incorrect Items Selected**: as the chart (below on the right) demonstrates, the number of incorrect selections, across all systems was very low (i.e. averages ranging from 0.25 - 1.5). This information does not effectively identify any clear advantages/disadvantages, in terms of error rates, for any particular system when compared to the others.

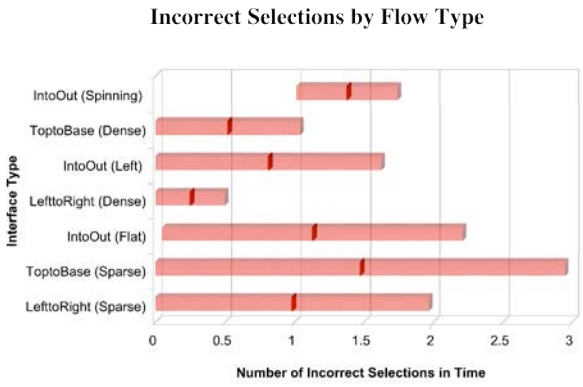



**Figure 7.11: Incorrect Selections by Flow type**

- **Number of Repeat Correct Items Selected**: As the number of repeat selections (i.e. double clicks rather than single clicks) was extremely low (only three cases across all users and all interfaces) it is not charted here.

Taking into account the combination of both positive (i.e. correct clicks) and negative (i.e. incorrect clicks/error rate) data the quantitative trial results clearly demonstrate the comparative effectiveness of the "IntoOut (Flat)" 3D presentation of the data. This interface incurred error rates along similar lines to the other systems while achieving by far the best positive numerical performance.

### *Qualitative Questionnaire Results*

The qualitative results for the flow trial represent the users answers to the following questions and provide an insight into the users ratings, preferences and opinions of the various interfaces.

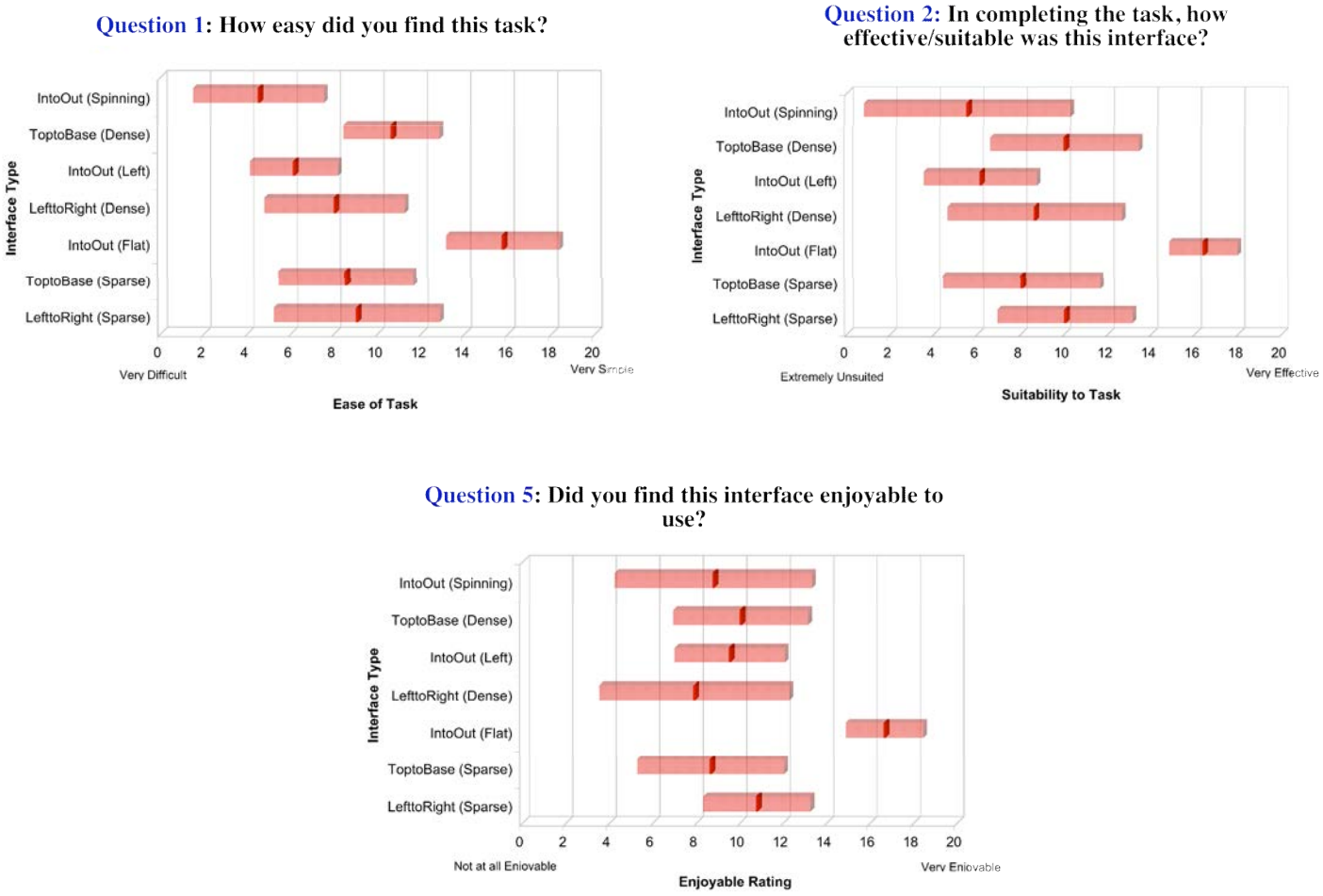



**Figure 7.12: Charts - User Assessment of Flow Interfaces.**

The charts above show the users assessment of the types of interface and their suitability to this selection task. It is clear that the “IntoOut (Flat)” interface is the most highly rated in terms of suitability to its task (i.e. Question 2). This value is also reflected in the fact that most users found the task simpler (i.e. Question 1) and more enjoyable (i.e. Question 5) using the “IntoOut (Flat)” interface. These user opinions match with the quantitative performance results outlined previously and indicate that the “IntoOut (Flat)” interface was both the most popular with users and also the most effective in achieving its task.

Of the other interfaces, the results fall broadly into a range in which no particular system stands out from the others. Based on the average results the “ToptoBase (Dense)” interface consistently comes in second, although some way behind, both in terms of numerical (i.e. quantitative) performance and user opinions. As the charts above show the remaining set of primarily 2D systems (i.e. ToptoBase and LefttoRight) are generally grouped relatively closely together in positions two to five.

The notably poor performers from these trials, consistently rating the worst of the interfaces, are the more complex versions of the 3D flow (i.e. “IntoOut-Left” and “IntoOut - Spinning”), these systems fail to perform in terms of both quantitative performance measures and user opinions.

The questions above primarily record the users rating of the interface in its task, however these trials are also interested in assessing the effect in terms of comfort, confidence and improved usability that these interfaces have on the users (i.e. what sense of understanding and control do users have with these systems?). The following questions provide an input in these areas:

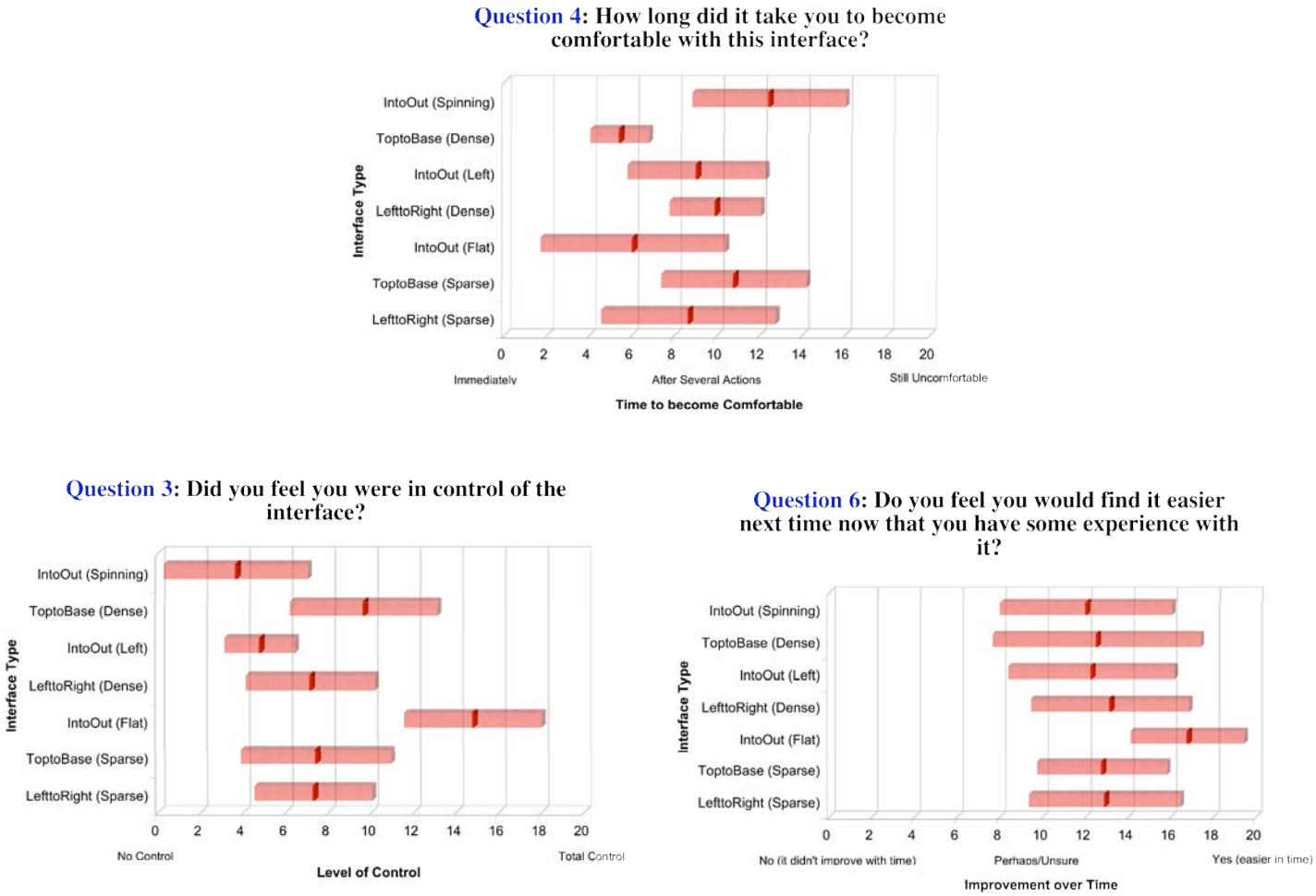



**Figure 7.13: Charts - User Comfort & Confidence Levels.**

The charts displayed on the previous page offer an interesting set of results, for example the users rated the fundamentally two dimensional “ToptoBase (Dense)” interface as being the most comfortable in the shortest time. In this example the lower the score the better the interfaces performance, with “ToptoBase (Dense)” generating the best score although not significantly better than the 3D “IntoOut (Flat)” system. Interestingly this is the first and only questionnaire or numerical result where the 3D “IntoOut (Flat)” interface was not the best performer. In this case the 3D “IntoOut (Flat)” came a close second and was statistically not significantly different. Behind these leading interfaces came the group of other essentially 2D systems. In addition in this result we also see the 3D “IntoOut-Left” interface performing better than in previous questions, with a result at a similar level (in fact slightly better) to that of the bulk of the 2D based systems.

As the other charts demonstrate the users found the 3D depth oriented “IntoOut (Flat)” interface to be the system they felt most in control of (i.e. Question 3) and also rated it as the most likely to become easier to use over time (i.e. Question 6). For the control oriented question, results of the other interfaces followed a similar pattern to that described in the earlier charts where the essentially 2D systems, led by the “ToptoBase (Dense)” interface, followed in a relatively tight group notably behind the “IntoOut (Flat)” interface. Interestingly “IntoOut (Flat)” was the only interface with a result range greater than 10.0 in the control test. This indicates that, with the exception of the “IntoOut (Flat)” interface, users found the other systems offered them an undesirable lack of control (i.e. low values). Given that the “flow controls” were removed from this test, this result was expected. However the surprisingly high score obtained by the “IntoOut (Flat)” system highlights its effectiveness. Once again the result set is rounded out by the more complex 3D systems (“IntoOut-Left” and “IntoOut-Spinning”) which proved to be the least effective in offering a sense of control.

The experience chart above shows that users in general are confident that all interfaces will be easier with more experience, as demonstrated by the generally higher averages across the board. With no particular system clearly rated as more likely than any other and the bulk of systems rated in a very similar range. The “IntoOut (Flat)” interface was again slightly ahead of the field and rated as the most likely to be easier to use next time.

In summarizing these user questionnaire results it is clear that the “IntoOut (Flat)” interface stood out on almost all of the result charts as the clear performance leader. It was preferred by users, both in terms of its suitability to the task and in terms of its ease of understanding and sense of control. Of the other systems, rarely did one stand out significantly from the remaining pack, although it was interesting to note that “ToptoBase (Dense)” was consistently the second best performed system. This includes being more successful than its competitive 2D systems (i.e. ToptoBase consistently lead, only slightly over LefttoRight). In addition it (ToptoBase) rated as the interface which users were most comfortable with the most quickly. Both of these results could be attributed to the users existing experience. In that the ToptoBase systems most closely resemble the widely used 2D GUI scrolling list, and as such the users found them slightly easier to adjust to than the other systems.

### *Summary Questionnaire Results*

At the completion of the trial (i.e. having used all trial systems and completed tasks and questionnaires for each) the user is asked to complete a final summary questionnaire. The results from this questionnaire are as follows:

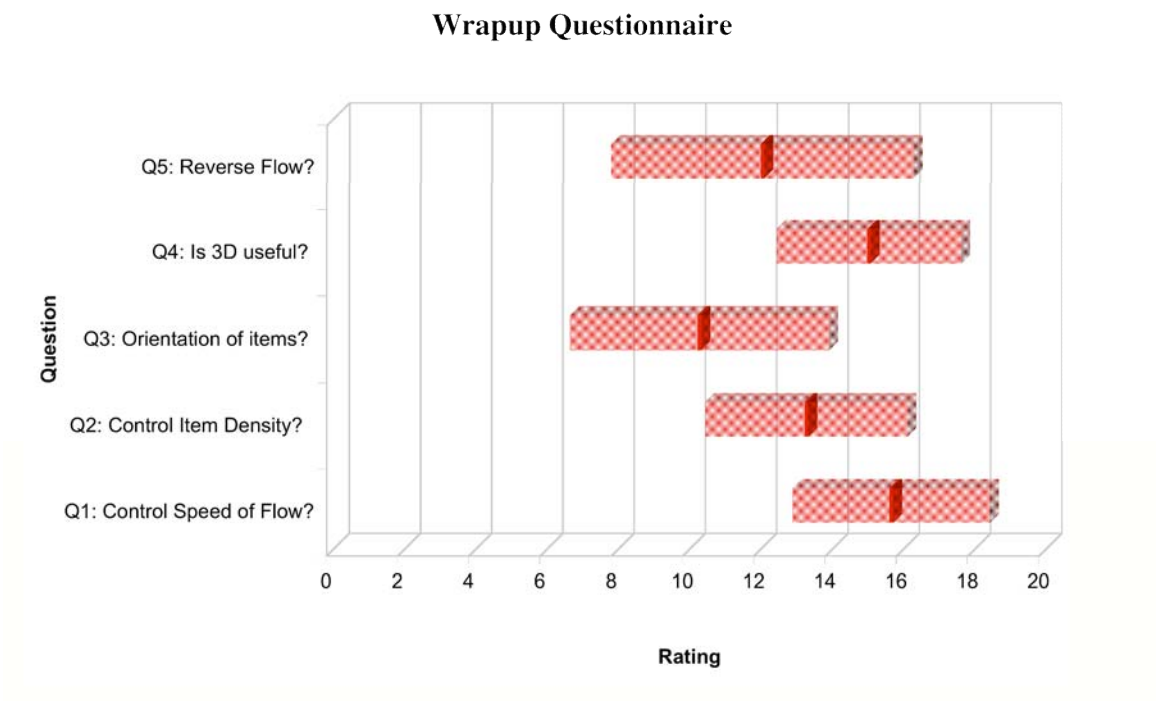



**Figure 7.14: Flow Component Wrap-up Questionnaire**

**Question 1: How much better/worse would these systems be if you could control the speed of the flow?** (0: Much Worse - 20: Much Better)

Responses to this question clearly indicated that users believe that all versions of the flow would be more effective with the addition of tools to control the flows speed. This result is matched by the fact that the earlier (system by system) questionnaires indicated that the trial users in general felt that they had less control of the interface than they desired. Clearly flow control is an important issue in implementing the flow component.

**Question 2: How much better/worse would these systems be if you could control the density of items in the flow?** (0: Much Worse - 20: Much Better)

The users believe the flows would be better, but not significantly so, given the ability to control the density of items within the flow itself. It was expected that this feature would be of similar importance to the ability to control the flows speed (especially given the varying levels of visual capability within the trial group). However this proved not to be the case as this “density control” feature was not rated as being as desirable/important as the ability to control the flows speed as described above. That said, the results do indicate

that 95% of users fall into the range above 10 on the scale, indicating that the flow would be better given the ability to control the item density.

**Question 3: In the last system the items were upside down (from your point of view). How important is the orientation of the individual items?** (0: Not Important - 20: Critical)

Based on user responses the orientation of the items in the flow is important but is not a critical element. In fact the user results do not clearly indicate that this is a necessary feature at all. Then results indicate that it is less important than many other factors. This result is surprising and demonstrates the users apparent ability to view the items somewhat independently of their orientation (i.e. able to mentally correct/upright items however they are presented).

**Question 4: Is 3D useful in presenting this type of data?** (0: Not Useful - 10: Same as Existing Methods - 20: Very Useful)

Given that the users were presented with a range of 3D interfaces, some of which proved very effective (eg. IntoOut (Flat)) and others which proved to be consistently ineffective (eg. IntoOut (Spinning)), their overall assessment of the usefulness of 3D for presenting this type of unstructured data was very positive. As the chart shows the broad range of user results falls at the high end off the usefulness scale, indicating that the bulk of users found 3D presentations to be more useful than any other existing methods (eg. 2D systems).

**Question 5: Would you like to be able to reverse the flow?** (0: Not Needed - 20: Yes, reverse is needed)

Results indicate that a reverse feature was desirable but not all that highly valued. It appears that most users would like to have the ability to reverse the flow but that the reverse capability is not a critical feature in making the flow interface effective in its task.

**Question 6: Do you have any other comments regarding any of the system used?**

Trial participants provided some interesting feedback at this question. These responses included both criticisms and suggestions for various interfaces. Responses to this question and the issues raised are discussed in the *Analysis & Discussion* section below.

## Analysis & Discussion

The key issue being assessed with the flow component is whether three dimensions, and in particular the use of depth, can be effective in presenting large set of unstructured data. The results as described above clearly demonstrate that a three dimensional depth oriented interface performs very effectively in this task. In fact the “IntoOut (Flat)” interface, which is essentially a flow coming from the depths of the screen, performed better in this task than any of the competing interfaces (including the fundamentally 2D systems ToptoBase and LefttoRight).

In analysing these results one of the issues raised involves assessing the impact of “experience” on the user results. The chart below shows the results from analysing a

particular case study involving an “experienced user” (one who rated their computer & 3D experience as greater than 15 out of 20) as compared to the “inexperienced user” (one who rated their experience as less than 5 out of 20) in terms of their quantitative selection task performance.

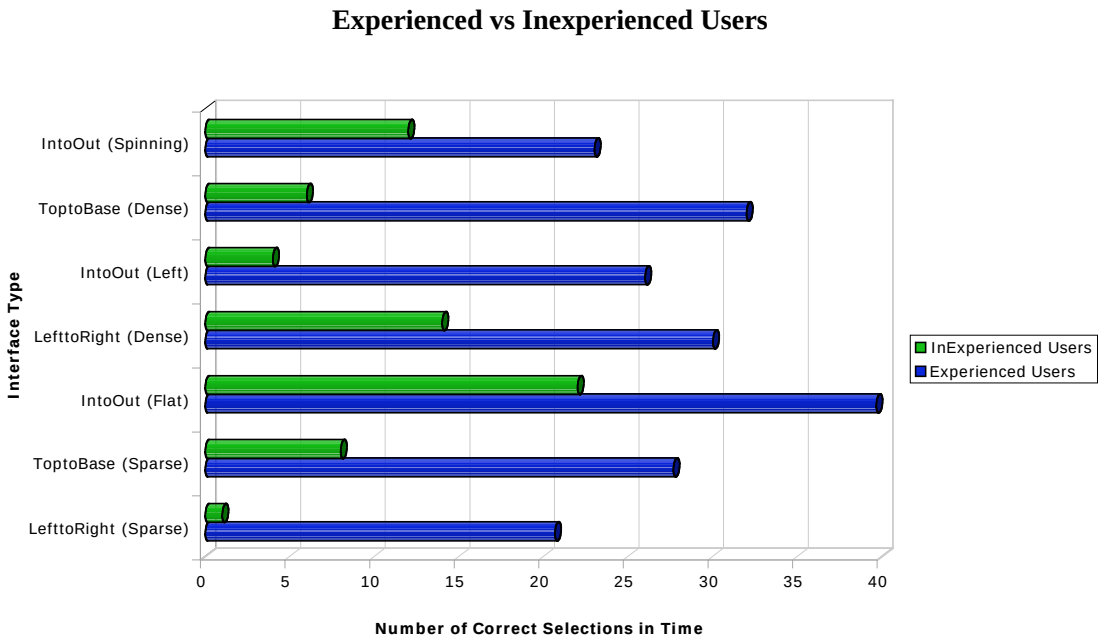



**Figure 7.15: Experienced vs. Inexperienced Users**

As the chart demonstrates, and would be expected, the inexperienced users perform less effectively in completing the task (in fact their results are far less consistent in general). However one interesting factor is the fact that the 3D “IntoOut (Flat)” interface remains the best performed interface for both experience levels. In both results it is (approximately) proportionally the same amount more effective than its nearest competitor. The fact that the 2D systems also appear to perform slightly less effectively (eg. ToptoBase is not the second best performer) for novice users may indicate that there is a bias towards these systems caused by existing experience. As interesting as this fact is in analysing these results, it is important to keep in mind that although “experience” may cause a bias, that bias is part of the representative population of trial users (and in fact is likely to be part of the broader population).

Another key difference observed between the interfaces involves the number of items visible at any one time. For example the IntoOut systems all display the entire set (i.e. all 640 items) on the screen at one time (this capability is a feature of the 3D system itself), whereas the 2D oriented systems (ToptoBase and LefttoRight) are limited in the number of items that are displayed at any one time. It is theoretically possible to fit all 640 items into the view at any one time in all systems, however to do so in the 2D systems would make the items so small that the system would be unworkable. The trials used two variants of each 2D system (Dense and Sparse) in order to provide items at two differing sizes and screen densities. In general the “Dense” systems (i.e. more items of smaller size) slightly outperformed the “Sparse” systems, although in most cases the differences

were minor. In fact it was generally the case that “ToptoBase (Dense)” out performed “LefttoRight (Dense)” even though LefttoRight has more items visible at one time (due to the width to height ratio of the view). This indicates that having more items visible does not directly generate better results. In addition to these raw values is the fact that one of the more common comments provided by users was to indicate that the items were too small in the “dense” variants. Given these factors, it is clear that providing the user with a capacity to control the size and density of items in the flow is a worthwhile feature (especially given the variety of user preferences).

Another related issue is the fact that the IntoOut systems are able to represent their data more effectively, yet use significantly less screen/view space. Again this is an advantage provided by the nature of the 3D depth representation.

Overall in analysing these results it is clear that depth can be effectively used for this task. However, as demonstrated by the “IntoOut - Left” and ”IntoOut - Spinning” interfaces, the simple inclusion of depth into a system does not guarantee its success. In both of these cases the interfaces performed poorly despite the fact that they used the depth of the screen to represent the set. These cases highlight the fact that excess motion does not improve interactivity. In fact many users commented to say that they liked the motion of the spinning system but found it useless for the task. Clearly “visually interesting” does not relate to “interactively effective”. The primary lesson here is that the simple system was the most effective. The “IntoOut (Flat)” system was the simplest of the fundamentally 3D (i.e. depth oriented systems) interfaces and this simple use of the depth provided by the 3D interface generated the best results.

## Flow Results Summary

The user responses to the summary question of “Is 3D useful for presenting this type of Data?” clearly indicated that the users felt that 3D is useful for presenting these large sets of unstructured data. This positive response is reflected in the performance results of the individual trial systems where the 3D “IntoOut (Flat)” interface was clearly the best performing interface both in terms of quantitative performance measures and qualitative user opinions. As such these results clearly indicate that a set of items flowing from the depths of the screen through and past the user is very effective, in fact more so than any of the other systems tested, in presenting this type of large unstructured set of data.

## Assessing the Orientation Aid Component

The primary purpose for the orientation aid is to help the user understand his/her orientation relative to the world in which he/she is located (see the earlier section on *The Orientation Aid* for a more thorough description of the component itself). The overall objective for this aid is to provide a visual cue (which can confer its information quickly and naturally to the user) to aid the users sense of their personal orientation in the 3D environment. This trial tests a range of differing "aid components" to assess how effective each is in providing the user with this sense of orientation.

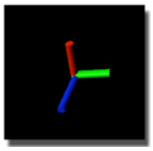
### *The Task*

The task undertaken by the users in this trial is to locate an item (hidden inside a box) following a reorientation of the world in which that box is located. In simple terms the user is shown a set of boxes in a 3D space. In one of these boxes a green ball (i.e. the target) is inserted. The user is able to observe the target being placed in the box and then observes the box being closed. The box is now solid and one of a set of identical appearing boxes all laid out inside this 3D space. When the user starts the trial the system automatically reorients his/her orientation relative to the world (implemented as a mechanical animated spinning process which spins the world relative to the user for a random but greater that 5 second period). The intent of this reorientation is to confuse the user and hence disorient him/her. Due to this disorienting process the user (without any help) should not be able to determine the location of the box containing the ball. Hence when the user attempts to identify the box (i.e. this being the task) he/she will essentially be guessing at the location. By introducing the various orientation aids into this task it is possible to comparatively determine which aids are more or less effective in helping the user reorient and thus locate the target box correctly.

### *The Trial Systems*

This trial tests eight different orientation aids in this task. The orientation aids that are being compared in this set of user tests are as follows:

No Aid - This trial starts with the user attempting to locate the target with no orientation aid at all. This gives the user no indication of orientation information.



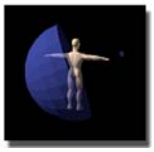
The Axis Device - This device spins on its central axis and indicates the positive directions in the X, Y and Z axes through the display of these coloured lines. This component provides orientation information in all three axes.

The Artificial Horizon - This device is based on the real world artificial horizon and displays a sphere (which rotates about its centre) with a blue (i.e. the sky) and a brown hemisphere (i.e. the ground). This device provides orientation cues in the one axis (i.e. up/down) but does not provide directional information).

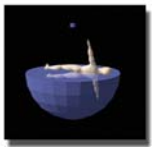
The Compass - The compass provides both a ground hemisphere and also a direction arrow/cone. thus this component provides both up/down and directional cues.

The Plane & Normal - Using a green ground plane and a normal (i.e. red arrow) this component provides a simple “up direction” only.

The Hand - This more complex device provides the user with a representation of the human hand. The fact that the hand is unique in differing axes/directions makes this component capable of providing visual cues in all axes.



The Man & World (world moves) - In this system the model of the man remains static and the world (or blue hemisphere) rotates around him to demonstrate the state of the world. Interestingly this component most accurately reflects the true state of items from the users perspective.



The Man & World (man moves) - Similar to the previous system however in this system the model of the man moves and the world remains static. This reflects the perspective that would be visible from an external viewpoint where it is clear that the user is reorienting in the static space.

***The Trial Itself***

The trial consists of the user undertaking the task of locating the target box in each of the nine differing systems (i.e. the eight described above and a repeat of the “no-aid” version to determine if the user is simply improving at the task over time rather than being aided by the components). The position of the target ball/box varies from trial to trial via a random placement of the target in the set. In addition the period of time during which the set is reoriented is randomly varied between trials to ensure that the user is presented with a unique set/orientation in each individual test.

It is difficult to guarantee the “disorientation” that this trial seeks to achieve, however it is possible to ensure that the user is not able to “visually follow” the target item during the reorientation phase, by reorienting in such a way that the set is turned away from the users view thus making it very difficult for the user to follow the items path. Essentially this trial records both the speed of the users response and the number of errors he/she makes before reaching the correct target and completing the trial. The quantitative data provides an understanding of how effective in terms of speed of understanding and accuracy of understanding the varied components are. In addition to the quantitative data recorded this trial also asks the users to complete a questionnaire following each test to indicate how useful/effective they found the orientation aid. The results from these trials follow in the next section.

## Results for the Orientation Aid Component

This trial was undertaken by ten participants, with each participant completing the selection task. At the start of the trial every user is asked to complete the task using the "no aid" system, this task is repeated at the last trial of the system. The objective from these two trials is to determine whether the user is improving at the task simply through experience (rather than through the help of the aids). The systems between are presented in random order. The charts below show the performance results of the users in both the first run with no aids and the second. Clearly the users ability to identify the target did not improve between the first and second runs (in fact the users were less effective i.e. made

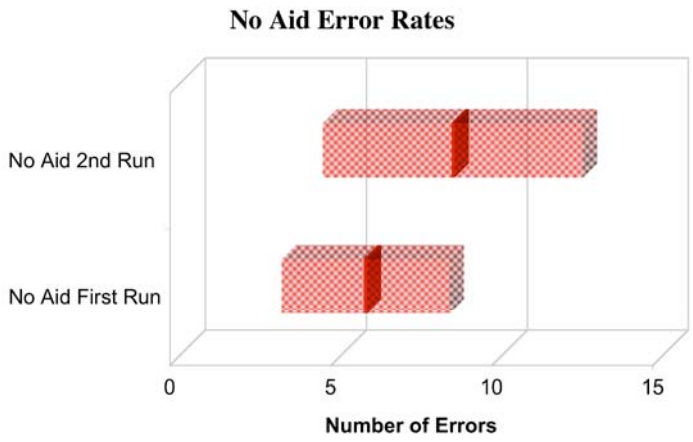



**Figure 7.16: Chart - Errors with "No Aid"**

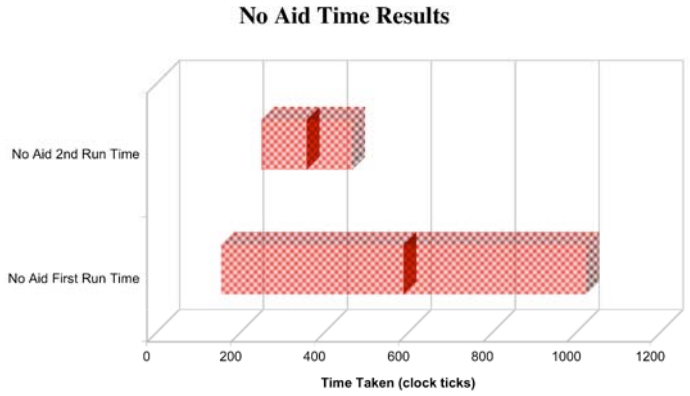



**Figure 7.17: Chart - Time with "No Aid"**

more errors the second time through). This is evidenced by the fact that the average number of errors increased in the second run results, as shown below.

The chart on the right above indicates that the users were, on average, taking less time to complete the task in the second run. However they were actually making more mistakes. This identifies the fact that the users were becoming more comfortable with the selection task and hence completing it more quickly, but they were no more skilled at determining their orientation or the location of the item over time. Hence the task itself does not get easier in time (although the users get faster at completing it) and we can judge the comparative effectiveness of the orientation aid components by assessing their performance in terms of error rates.

The following chart displays the comparative performance results, in terms of error rates for each of the orientation aid components. The large ranges in many of the error rate values indicates that although some users liked a given device, others found it totally ineffective. Interestingly it is two of the simpler devices that stand out as the best performers. The "compass" device with its simple ground hemisphere and direction arrow proved effective as did the very simple "plane and normal" device, which is similar in style consisting of a ground plane and an up arrow. The more complex devices (eg. the hand, man etc) proved to be less effective than was envisaged. Thus identifying the fact that for the task of supplying simple orientation cues, users prefer the simpler more obvious components.

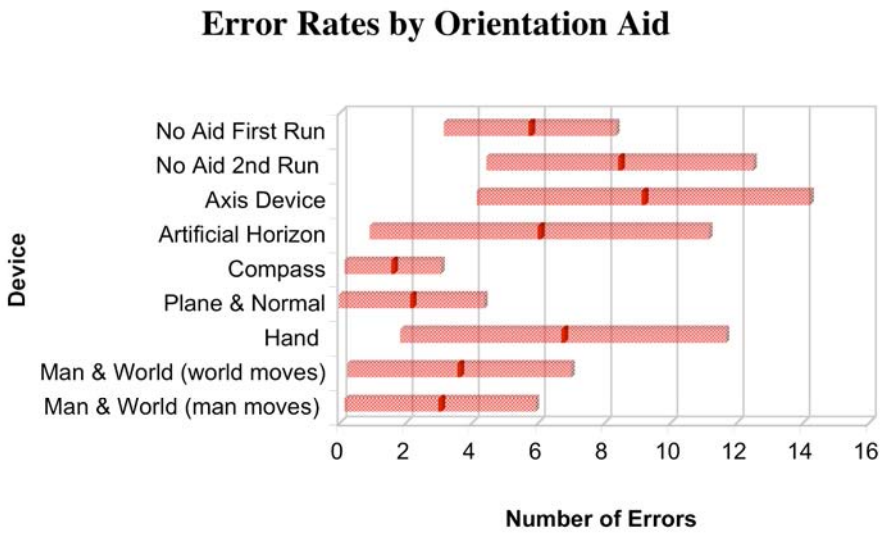



**Figure 7.18: Chart - Comparative Orientation Aids.**

These performance results are also largely reflected in the questionnaire responses of the trial participants with the "compass" and "plane & normal" devices performing well in user question responses. Unfortunately the user responses cover very broad ranges and as such these qualitative results indicate little in terms of identifying one system as superior (or otherwise). Below are charts showing the results from the key questions of: "How disoriented did users feel? (0-not at all to 20-very disoriented)", "How confident were they of their orientation/selection?" and "How effective was this device in its task?".

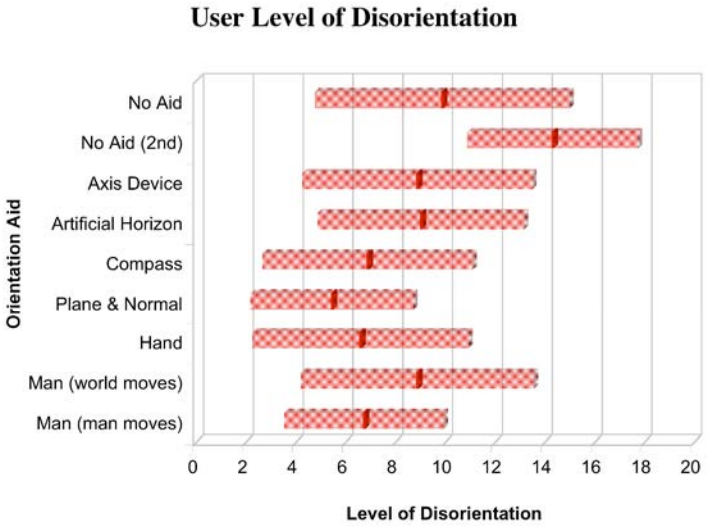



**Figure 7.19: Chart - Disorientation Level**

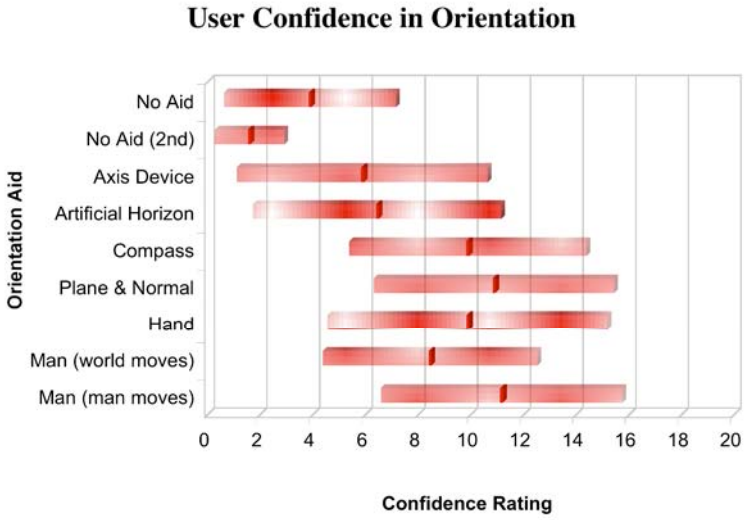



**Figure 7.20: Chart - Orientation Confidence**

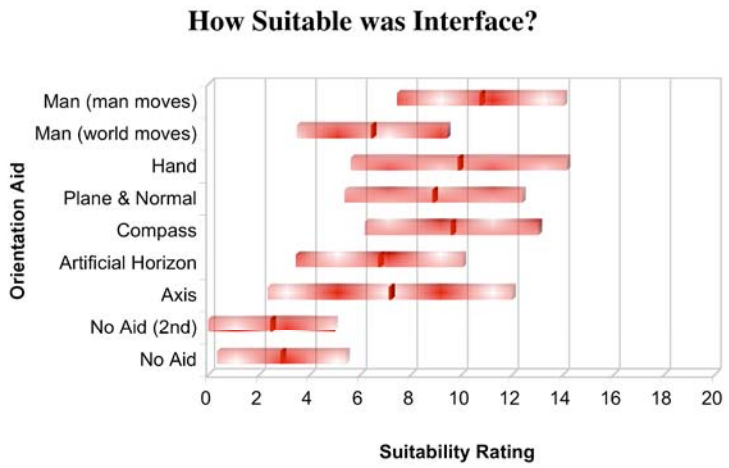
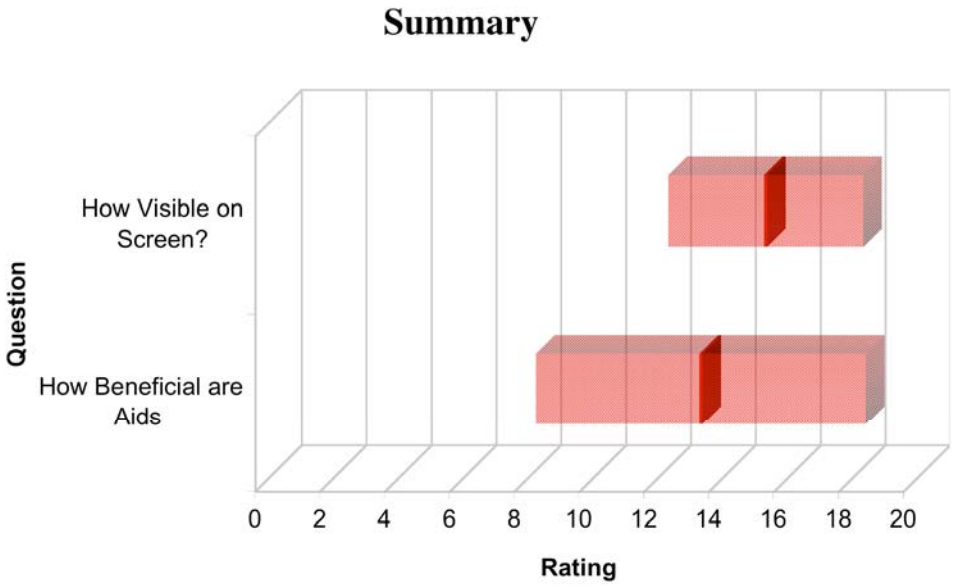



**Figure 7.21: Chart - How Suitable were Orientation Aids**

From all of these results it is clear that the level of user variation in this task was large and hence it is difficult to identify any clear winners in terms of effectiveness (although in all results it is clear that the "no aid" systems were inferior to aids of any kind). Of the entire set only a couple of simple devices appear to be universally popular with the users. Those being the "Compass" and the "Plane & Normal". Interestingly all of the devices rated relatively poorly on the suitability question, indicating that perhaps an orientation indicating tool of this kind is more difficult to implement effectively than initially thought (see a discussion of this issue in the earlier descriptive *Orientation Aid* section of the thesis).




At the conclusion of the trial all users answered a set of summary questions asking about their preferred aids and how valuable they felt the aids were overall. As the chart shows most users felt that having the aid visible at all times was useful (although this was taken immediately after an orientation based trial and it would be interesting to see if they had

the same feelings after other less orientation based activities). On the question of their preferred aids, the users rated the “Plane & Normal” device as their favoured option with 33% voting it their first choice amongst the set presented. Overall the users indicated that the orientation aids were beneficial (see previous chart). This result is reflected in the performance figures where it is clear that the aids (some more than others) clearly improved the users performances.

## Analysis & Discussion

The key issue that these results identified is the fact that the simple systems (i.e. Plane & Normal and Compass) outperformed their more complex equivalents. Evidently simple is best when it comes to indicating orientation to the user. This is particularly interesting as it had been hoped that the more complex systems (i.e. particularly those which contained more information) would prove effective, however it seems that the additional information simply made the aids too complex to be utilized or understood effectively by the user. The very simple up arrow of the Plane & Normal device provided the users with the information they wanted in the simplest and hence most popular form.

## Orientation Aid Results Summary

These results clearly identify that the user can perform better (in terms of understanding their orientation in space) when an orientation aid is present. However the particular details of that aid make a significant difference in the effectiveness of the user in these tasks. Clearly simple devices that demonstrate a single clear piece of information (eg. the Plane & Normal device) are more effective than those that are more complex and display more details. Despite the fact that the man based system provided more information about the users state relative to the world he/she was in, they proved to be less effective and less popular with the users. Hence in summary, the Plane & Normal was the device preferred by the users. It also featured as the second most effective device in its task (behind the Compass) and was interestingly more effective than the widely used (i.e. real-world) artificial horizon device.

## Assessing the 3D Slider Component

The 3D slider presents a collection of new interface components for the task of setting values within fixed limited ranges. For more detailed information on these components see the earlier section *The 3D Slider* and also the section *The Simpler Components-Line Slider*. To assess the effectiveness of these components each is tested in the task of value setting by utilizing it to set a specific value. When this same task is undertaken with the range of new interfaces these values can be comparatively assessed to determine which components are more or less effective.

### *The Task*

The task that is implemented by the trial participants involves using the "3D slider" component (as shown on the right below) to set a value (in this case a scale value that enlarges/reduces the size of an item in space) so that the resulting item is as closely matched to the original as possible.

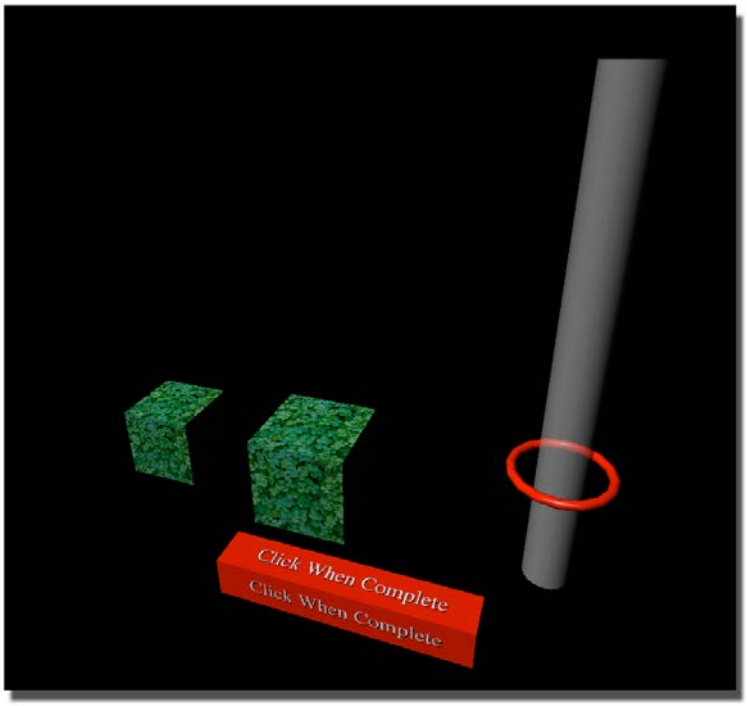



**Figure 7.22: Screenshot of the sizing task (Line-Slider)**

Thus the users are required to use the "sizing component" to set the size of an item so that it is equal in size to another item in the world (i.e. make the green box, on the right, match the green box on the left). In completing this task the user must utilize the features of the 3D slider to enlarge or reduce the size of the item. The objective is for the user to get the two items as close in size as possible in the shortest time. This trial gives quantitative measures of both speed of performance and also accuracy. In addition to these quantitative measures the users also answered a series of questionnaires to determine which components he/she finds the most effective and enjoyable.

### *The Trial Systems*

The task itself remains consistent across the trials, however the tool that is used to set the size is varied from trial to trial. Hence the user has the opportunity to use all of the trial systems to complete the same task. The tools/components that the user completes the trial with include the following:

*The Line-Slider* (Front & Top Views)

The line slider enables the user to drag the red ring up and down along the tube to set the value in the range between the top of the tube (max. size) and the base of the tube (min. size). Essentially a 3D version of the basic 2D slider, this interface component is presented in two variants, each displaying the component from a different viewing angle. The first is the front view where the Line-Slider extends along the Y axis of the user's'view. The second is the top view where the Line-Slider extends along the Z axis of the user's view (i.e. extends into the screen).

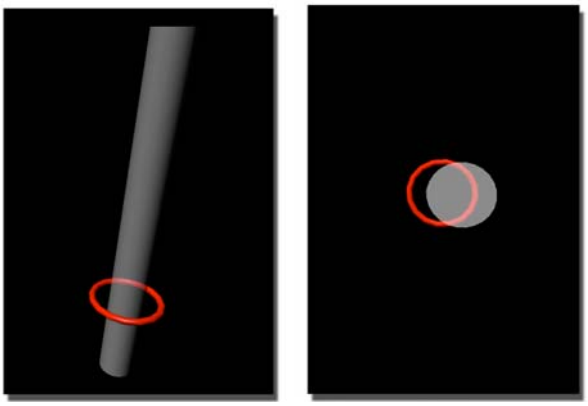


**Figure 7.23: Line Sliders (Front & Top)**

*The Sphere-Slider*

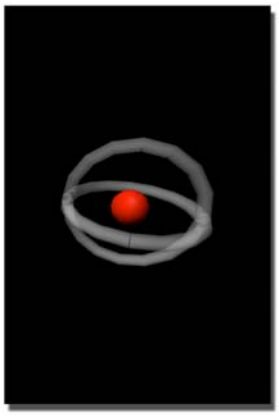


**Figure 7.24: Sphere Slider**

This component allows the user to drag the red ball to enlarge/reduce. The direction of the drag determines if the ball enlarges or reduces in volume (i.e. a drag towards to centre of the component reduces the size; where a drag away from the centre enlarges the size). As the user drags the red ball increases in size ranging from the minimum value at the centre of the linked rings to the maximum value where the volume completely fills space defined by the linked rings.

*The Spring-Slider*

The spring slider enables the user to drag the red ball up and down along the curved tube to set the value in the range between the top of the tube (max. size) and the base of the tube (min. size). Very similar in style to the Line-Slider, however the 3D curve of the tube makes this component capable of viewing from all angles.

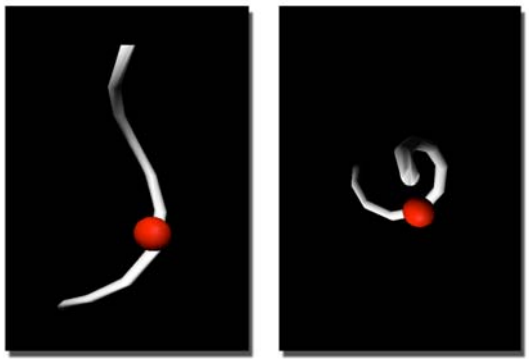


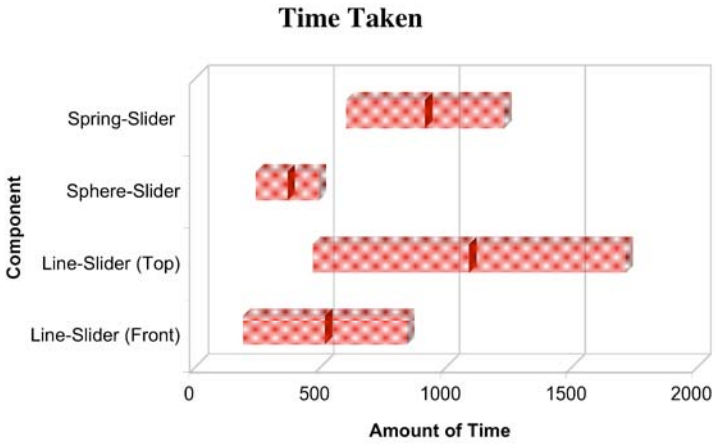
**Figure 7.25 Spring-Sliders (Front & Top)**

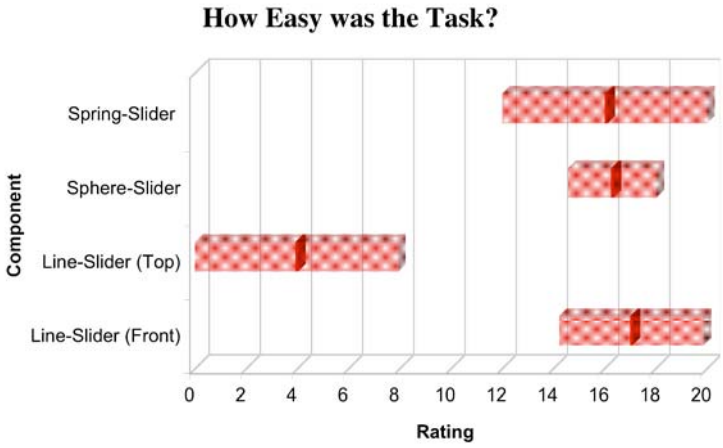
***The Trial Itself***

The trial simply involves presenting the user with two objects (i.e. one being the target size and the other the item to be resized). Located beside those items is the resizing device (i.e. one of the devices shown above), which can be used to resize the item to match its target. While the user is undertaking this task the time taken and accuracy of the final results are recorded. When the user completes each resize task he/she is presented with a questionnaire asking how effective this component was in its task as well as a series of other questions. The user is then transferred to the next resize task/component. At the completion of the tasks the user fills in the summary questionnaire which asks questions such as "Which component is most effective for this task?" as well as which is least effective. The results from both the quantitative and qualitative data from these trials are outlined below.

## Results for 3D Slider Component

This trial was undertaken by six participants, with each participant completing the full set of tasks. The order of presentation, which was consistent throughout the trial is outlined in the result charts below, with the bottommost system being presented first. Results from the slider trials provide an interesting insight into the effectiveness of the various systems being tested. It is a known fact that the Line-Slider in its frontal presentation is a well established existing system (i.e. the 2D Slider) and as such will benefit from the fact that



users already have experience with it. However despite this advantage the results indicate that the users found all but one of the systems to be relatively equal in terms of their effectiveness in the value setting task.

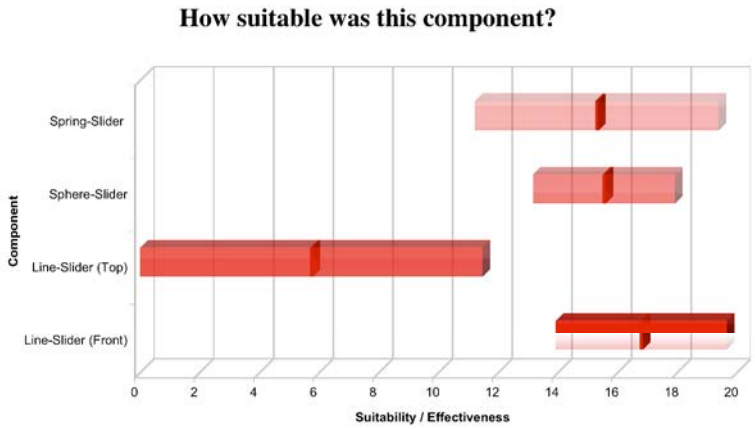
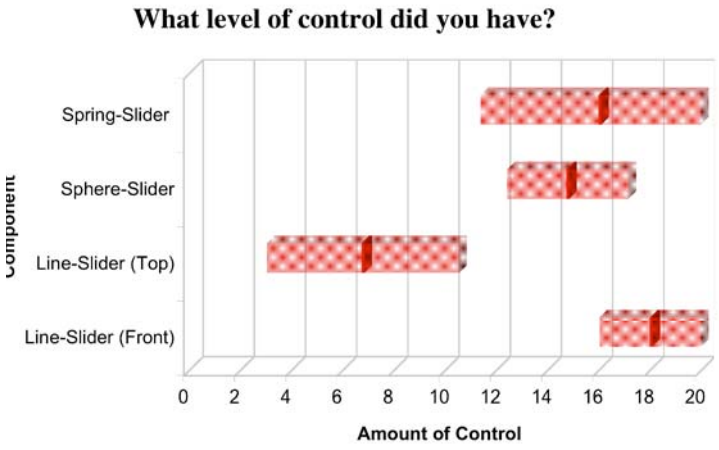
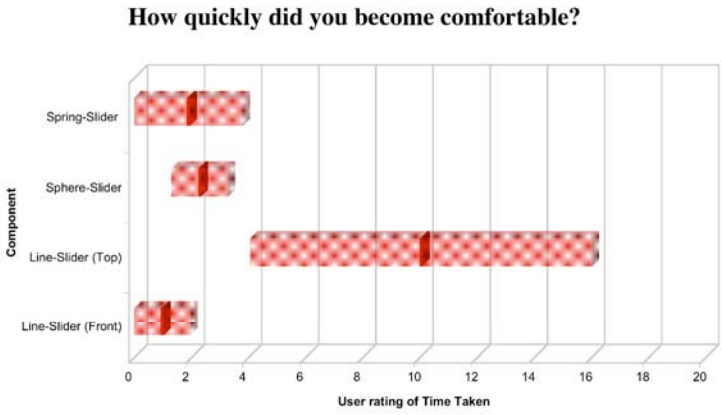



**Figure 7.26: Charts - Effectiveness of 3D slider components.**

As the charts show both the quantitative assessments and the qualitative user responses indicate that the Line-Slider (Top) is unsuited to its task. Aside from this fact the other three systems performed relatively equally. Surprisingly users found the volume based Sphere-Slider to be the fastest (although it is not significantly different to the others). In general these results indicate that the Sphere-Slider, Line-Slider(Front) and Spring-Slider are very similar in terms of their effectiveness.

Other recorded data also indicates this with the sizing accuracy (which is not shown here as the systems were all essentially equal) indicating that all systems were equally capable of accurately setting the items size to match its target. In addition to the performance measures users also indicated how comfortable and enjoyable the systems were. These results are shown below.



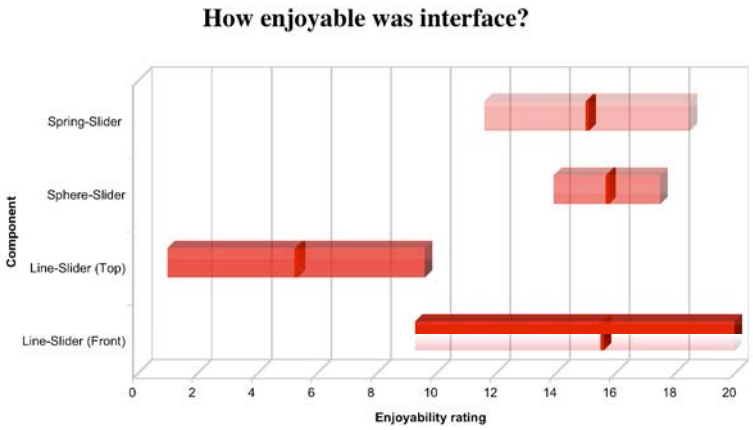



**Figure 7.27: Charts - The users opinions of the sliders.**

As the charts above indicate the users found the Line-Slider(Front) slightly easier to control and also became more comfortable with it more quickly. This is most likely due to their existing experience with the component in its 2D form. Overall its advantage in all these categories is very small and as such it is clear that users found all three (Sphere, Spring and Line(Front)) equally easy and enjoyable to use.

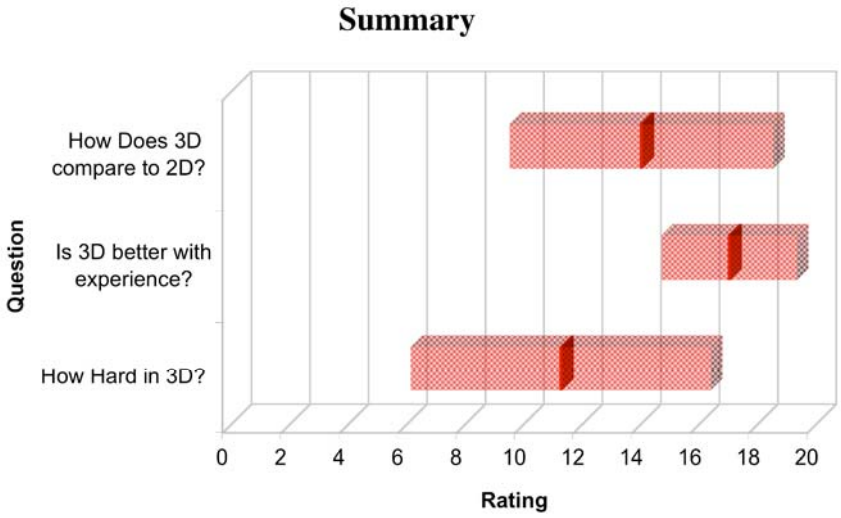



**Figure 7.28: Chart - Summary of 3D sliders.**

In summary the results from the users “wrap-up” questionnaire indicate that the participants found the task equally difficult in three dimensions as it is in two dimensions (i.e. the task did not get easier in 3D). They believed that the 3D components would be easier to use with more experience and they rated the 3D components as being slightly better than their 2D equivalents. This last result most likely reflects the fact that the 3D systems can be truly imbedded into a 3D space where the 2D systems can only be overlaid.

When asked which sliders were most effective 44% indicated that the Sphere-Slider was their preferred option another 44% also indicated that the Line-Slider(Front) was preferred and 12% indicated the Spring-Slider as their first choice. When indicating their

least preferred option 100% of the trial users indicated the Line-Slider(Top) interface as the component they would least like to use for this task. These results are interesting because the Line-Slider is both the most popular (when viewed from the front) and the least popular (when viewed from the top). Clearly this result shows the key problem that many 2D based components have when transferred to 3D systems. Although they are effective in 2D they do not work well in 3D environments. However components such as the Spring and Sphere sliders, which are specifically designed for 3D space are equally effective from all angles and as such are more suited to application in 3D interfaces. The fact that these results indicate that the users find these new 3D components equally effective, with their 2D equivalents, represents a positive step forward for the specification and implementation of fundamental new 3D interface components that can effectively replace the well established 2D systems.

## Analysis & Discussion

The particular task undertaken in this trial is a sizing task. This type of task closely matches the style of interface presented by the volume based Sphere-Slider and as such this component may benefit from the nature of the task itself. Although this is true, it is also true that a great proportion of value setting tasks fall into this type of category (eg. setting colour intensities, setting range values, strengths etc.). Clearly the Line and Spring sliders are more general in their presentation style and may be more suited to general value setting tasks. However, by providing both the Spring and Sphere slider this offers the developer the option of selecting the value setting mechanism that is most suited to his/her particular task.

## 3D Slider Results Summary

These results clearly show that the Line-Slider (i.e. basically a copy of the 2D slider) remains effective in 3D only when viewed from a frontal (i.e. square on) viewing perspective. When viewed from above this component becomes ineffective in its task.

The results indicate that the newly created Sphere and Spring slider components (developed by this research) are able to function as effectively as the Line-Slider (when viewed from its optimum angle) and also offer their functionality when viewed from all angles. Making them more suited to use in 3D interfaces. The new components created in this research identify the fact that this critical task can be set using these 3D components as effectively as it can with existing 2D systems. This is a particularly pleasing result. As such these new 3D components can provide functionality that enables them to be embedded into a 3D space, viewed and function effectively from multiple viewing angles and locations, thus making them particularly suited to multi-user environments.

## Assessing the Drum Selector Component

This section outlines the user trials (and results) undertaken to test the collection of components designed for the task of selection from small sets of choices. This task is very successfully handled in 2D systems by the menu component and these trials seek to compare the menu based system with some new selection devices. The trials outlined below describe three interfaces, two of these are based on the menu with the other a new 3D component, the "drum". For a more detailed description of the workings of the Drum component see the earlier section *The Drum Component.* For more detail on the menu based components see the *Simpler Components* section. The following sections outline the details of the user test implementation and the results from those trials.

### *The Task*

To assess these simple selection components the user is required to make a particular choice from within the set. During this task the users performance, in terms of time taken and number of errors is recorded. The task itself involves the user choosing one item from a set of either eight (small set as shown below) or sixteen (medium set) choices as presented by the component currently under trial. Each choice is a textual word in the form of "Choice A", texture mapped to the side of the selection items as shown in diagram below.[32]

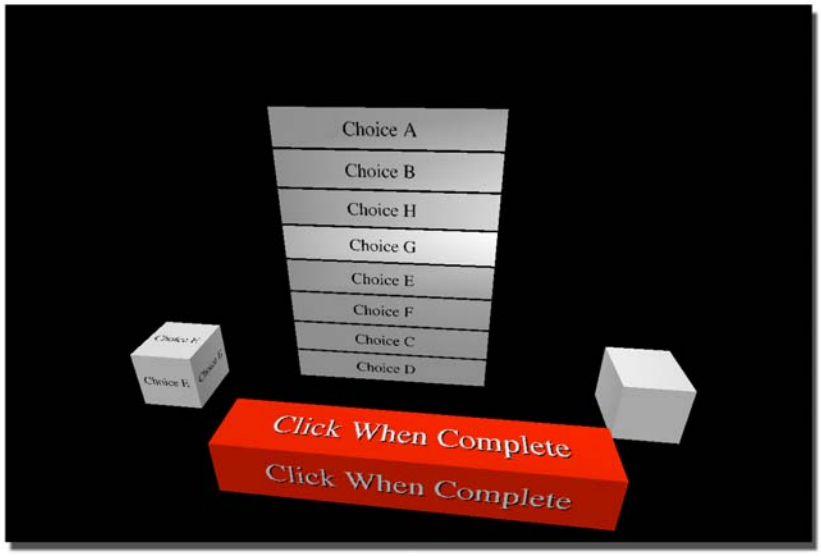



**Figure 7.29: Screenshot of the small selection set trials.**

The users task is to match the box on the right with the box on the left by selecting a target from the "selection tool" in the centre. The "target" item is randomly varied between component trials as is the arrangement of the choice options. In completing the

[32] For the task single letter differences were used. In real applications each item would be significantly different from the others and this would improve selection performance in all systems.

full test the user is exposed to a total of 12 systems incorporating the three components presented from differing perspectives (i.e. top, side and front) and with differing selection set sizes (i.e. small and medium). After making his/her selection with each tool the user is required to complete a questionnaire about the component used. These results in combination with the wrap-up questionnaire at the end of the trial provide the set of qualitative data for this trial.

### *The Trial Systems*

Essentially there are three systems being compared here, the “flat menu”, “box menu” and “drum” component. Each is designed to present small sets of data to the user for selection. In these trials these components are presented to the user in four forms including:

- *Front View* (Small set - 8 items): this presentation shows the component in its (preferred) front view orientation.
- *Side View* (Small set - 8 items): this presentation shows the component as viewed from the side. The side angle is slightly offset from exactly 90 degrees to enable the rectangle based “flat menu” to be visible from this angle.
- *Top View* (Small set - 8 items) ): this presentation shows the component as viewed from above. As for the previous system the viewing angle is slightly offset from exactly 90 degrees to enable the rectangle based “flat menu” to be visible from this viewpoint.
- *Front View* (Medium Set - 16 items): this presentation is almost identical to Front View (8), in that it shows the component in its (preferred) front view orientation. However the set of choices is larger.

**Flat-Menu (Rectangle)** in Top, Side & Front arrangements

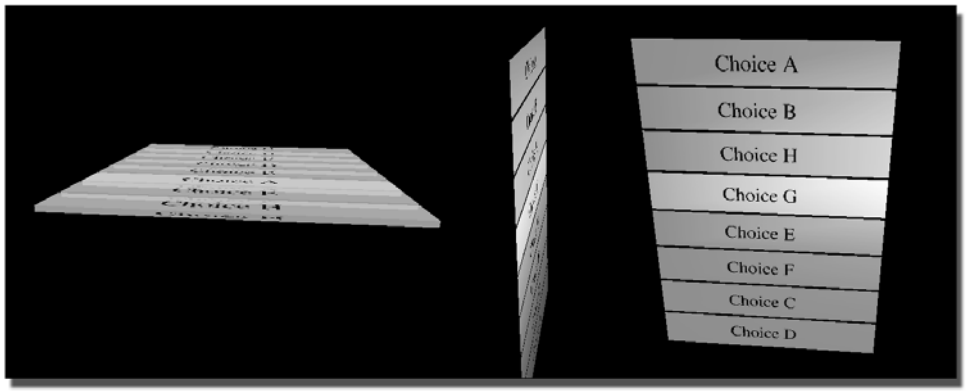



**Figure 7.30: The Flat-Menu (Rectangle).**

The flat menu is essentially a 2D rectangle based menu inserted into a 3D space by mapping the menu choices onto the faces of flat rectangular planes as shown in the diagram above. This component is highly two dimensional and provides a comparison system for the 3D systems to come (i.e. it is primarily there for comparison with an interface that is identical to a 2D menu).

**Box-Menu (Rectangular Prism)** in Top, Side & Front arrangements

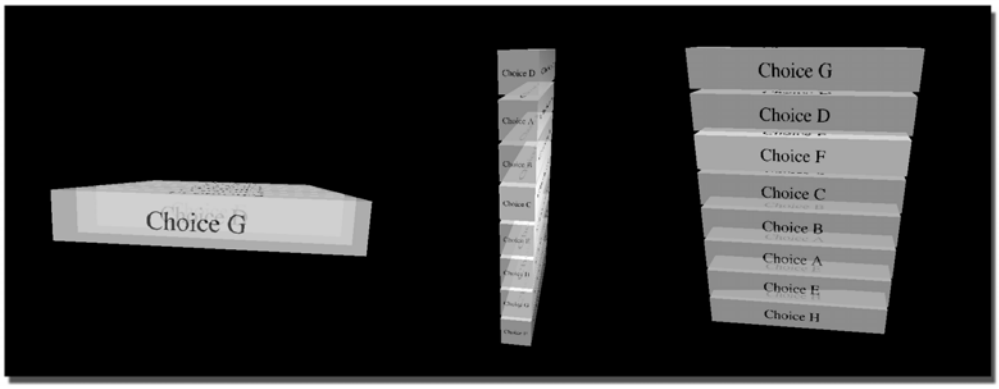



**Figure 7.31: The Box-Menu (Box).**

The box menu is a simple extension of the flat menu. It involves the use of rectangular prisms instead of flat 2D rectangles (as shown in diagram above). This component is more three dimensional than its predecessor however it is still fundamentally based on the 2D menu component. Its additional depth should make it more effective in 3D space, although it is not designed with multiple viewing angles in mind.

**Drum** in Top, Side & Front arrangements

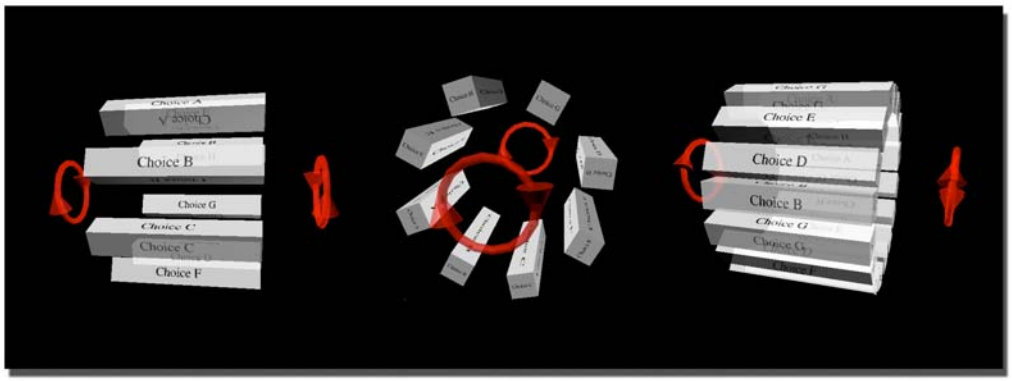



**Figure 7.32: The Drum.**

The drum is an entirely new component, specifically designed for the implementation of this selection task in a 3D space. It utilizes a cylindrical ring/drum of choices which can be rotated about their axis to cycle through the choices (as shown above). Intended to work when viewed from any angle this component provides the main innovation in this trial. Its performance, when compared to the menu based systems provides the key results from this trial.

## Results for the Drum Selector Component

This trial was undertaken by eleven participants, with each participant completing the full set of tasks. The charts that follow indicate the order in which the trial systems were presented to the user, with the first system at the base of the chart results. The results from this trial are in two key forms, the first is the quantitative set of results including the accuracy and time taken for the selection task. The second set of data is the opinions of

the users in terms of how suitable and enjoyable the component was in achieving its task. In terms of accuracy the results indicate that all users were able to correctly locate the desired target in all systems. Despite the fact that all the users correctly found the target at completion, some systems caused more incorrect selections (i.e. prior to final setting) than others. The chart below demonstrates the error rates for the various systems.

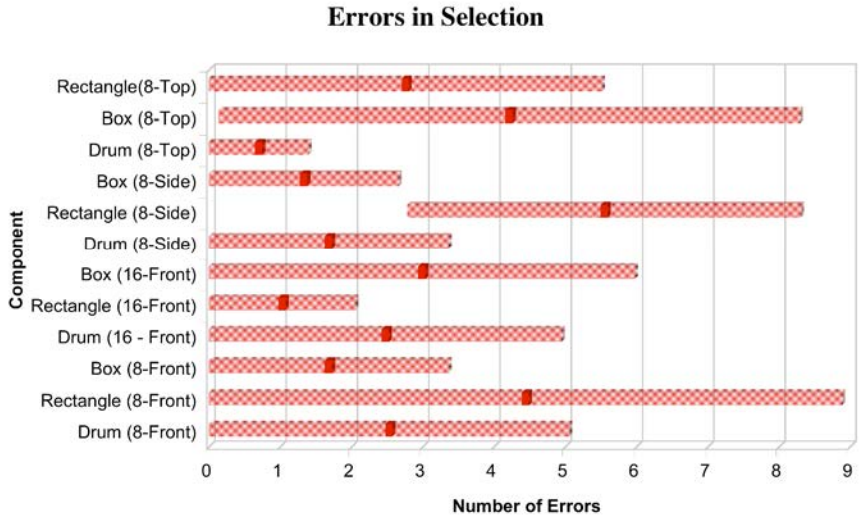



**Figure 7.33: Chart - Error rates for small set selectors.**

Although the broad ranges of many of the results in this chart make it difficult to identify clear winners or losers in terms of error rates it is clear that none of the systems are significantly better than the others in terms of errors. Minor variations can be noted, including the fact that for the “Top view” systems the drum average is better than its competitors. Equally it is clear that the rectangle based system is worse in the side view orientation.

Another key quantitative assessment is that of the time taken to complete the selection task with each system. The chart below shows the users results in this area.

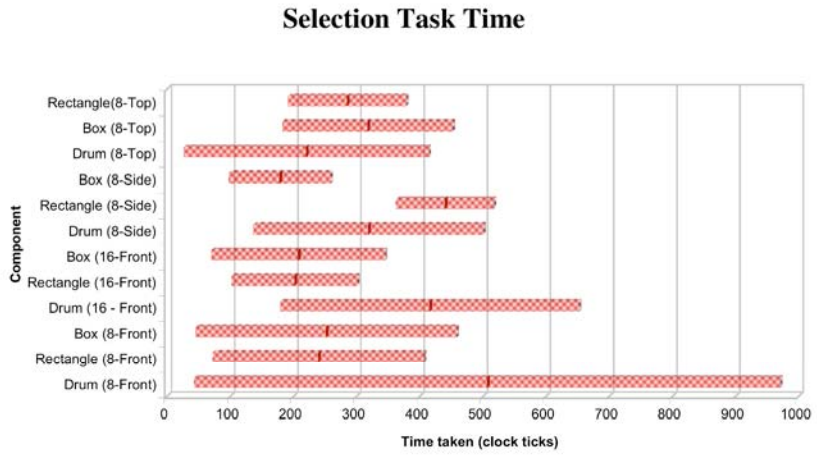



**Figure 7.34: Chart - Time taken for small set selectors.**

The most notable element in this chart is the fact that all of the drum component systems have very broad ranges when compared to the other systems. This could be due to the fact that the users are experienced with menu based systems and hence are relatively comfortable with them thus making the user responses more consistent for the known systems and less consistent for the new drum component. Aside from this fact the chart indicates that in general the rectangle and box based systems are faster when used in their frontal orientation (i.e. their optimum orientation) than the drum based systems. However when the view is shifted to the top and side views the drum is more effective. Interestingly the box menu (through use of its depth) is effective in the side view where the rectangle is not. Both of these menu based systems are less effective than the drum in the top view. Overall this chart indicates that the drum component is generally slower to use than its menu based equivalents, however it offers the ability of functioning from all angles (which the others do not achieve).

The users qualitative assessment of the components suitability to its task provides a similar set of results (see Figure 7.35), however it more clearly indicates the fact that the box and rectangle are unsuited to top view presentations.

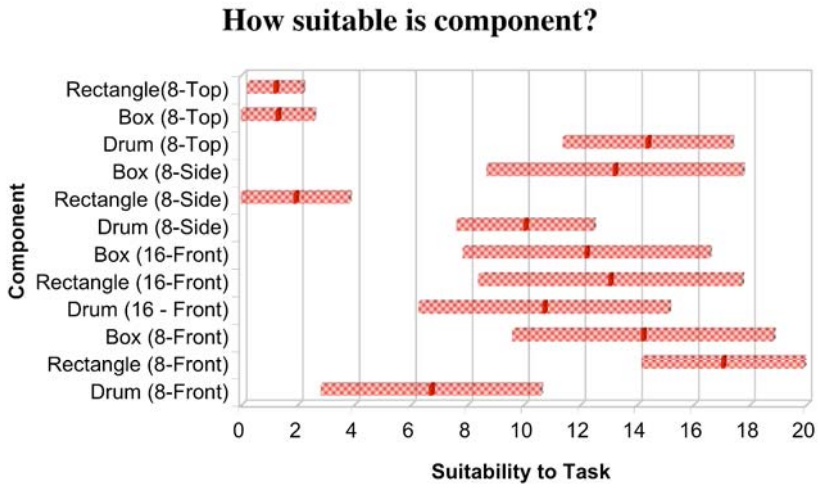



**Figure 7.35: Chart- Suitability to task for small set selectors.**

Other qualitative questionnaire results generally reflect this set of results (i.e. where the rectangle and box are effective in the frontal presentation but not capable in side and top orientations). The most notable difference from this basic response is shown in the users rating of how enjoyable the component was to use (as shown in the following chart). In this data the drum component leads in terms of popularity in four of the five presentation styles, with only the small set frontal arrangement falling slightly behind the others. This indicates that although the drum component was slower in completing the task, the users found it more enjoyable to use. In terms of the other two systems their ratings were very similar to each other; with one exception (i.e. the side view) where the box was clearly preferred to the rectangle. This identifies the relatively obvious fact that the rectangle is unable to function effectively from either the top or side view. The box, however, is only ineffective from the top view.

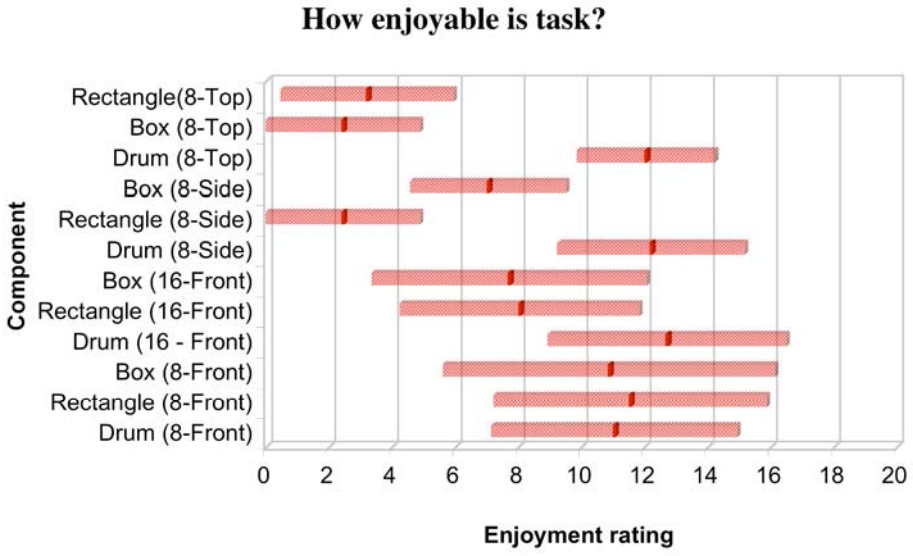



**Figure 7.36: Chart - How enjoyable are small set selectors.**

The final qualitative data obtained in this trial is the responses that the users made to the wrap up questionnaire. This questionnaire is designed to get broad overall answers and determine if certain design concepts (eg. transparency) are effective. The questionnaire in this trial asked users the following:

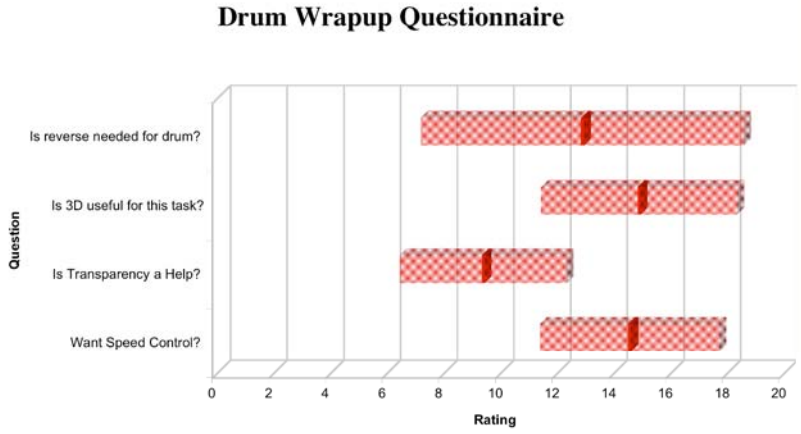



**Figure 7.37: Chart - Wrapup Questionnaire.**

As the results show the users are unsure if reverse would be a useful feature for the drum, however it is favoured by more users than disliked. Most users indicated that they believe that 3D is useful for this small set selection task. They were, however, unconvinced as to the value of the partial transparency used in the drum and other selection tools. The ability to control the drums rotation speed was desirable, but not a critical feature.

## Analysis & Discussion

One interesting result is the fact that the drum components "user suitability rating" was higher for the 16 item case (i.e. the medium sized set) than it was for the 8 item case (i.e. the small set). Perhaps this indicates that the drum is more suited to a medium sized set of data or perhaps this simply indicates the fact that the menu based systems are comparatively less effective with larger sets. The benefit of the drums more efficient use of screen space may make it more suited to sets in the 10-20 item range where the menu is more suited to the 4-10 item range.

Another unexpected result was the variation in performance between the top and front view presentations of the drum. In theory both the top and front presentations should generate very similar results, due to their similarity in form and presentation. The trial results indicate that users generally found the top view of the drum more suitable for the task. This may be due to the order of presentation (i.e. the top is presented first) or perhaps user fatigue (i.e. the later front system may be rated lower as the user becomes more fatigued). This does not effect the results in terms of the drums relative performance to other systems, however it is an interesting effect and may demonstrate that the "novelty" of the drum reduces over time.

The other key issue that effects these user results is the value of experience. Clearly all users have existing experience with menu based systems and as such the results are skewed towards users performing better with those systems. Interestingly the users indicated this fact when asked how much easier the task would be now they have experience with the system. Results from this question indicated that the drum systems would improve more than the menu based systems. When a subset of the users were re-tested with this system the results did indicate an improvement with the drum systems, however the overall performance when comparing the systems remained similar, with the drum improving slightly but not enough to change its overall status.

## Drum Selector Results Summary

Unfortunately the results from these trials provide inconclusive data in terms of specifying which components are more or less effective for the general task of small set selections. Clearly all of the systems are functionally capable, however determining one that is better or worse than the others is difficult and is very dependant on the particular application.

The user responses fall into broad ranges and are very different based on the presentation angles and consequently different components were more effective in different scenarios. Despite this the data does indicate that the menu based systems (i.e. Flat-Menu (Rectangle) and Box-Menu (Box)) are more effective when used in their preferred (i.e. frontal) orientation. Unfortunately these menu based components become ineffective when viewed from other angles. The drum component on the other hand is capable in all orientations however it is generally slower to use in completing the task. Despite the slower nature of the drum it was described as more enjoyable by users.

In summary the results from these trials tell us that the drum component created in this research is slower in completing its task and takes more time to become accustomed to. However the users believe they would speed up with experience.

The users found it functionally more effective from all angles and found it more enjoyable to use than the menu based systems.

## Assessing the Environment Interface ("Move-with-User" Tools)

This user trial actually incorporates the testing of a collection of components (i.e. the tool-belt, portal, pouch, Swiss army knife and orientation aids). These components are designed to be used together and as such the primary objective for these trials is to determine the effectiveness of the tool-belt and associated "move-with-user" tools. For a full description of each of these components, their design and implementation see the earlier section *'The Full Environment Interface'*.

To effectively assess the capabilities of components of this type presents some difficult challenges. Essentially the tool-belt and associated "move-with-user" items are most relevant to a user when they are undertaking other activities, for example navigating worlds, or on the web (i.e. it is a support component rather than a central component, supporting other tasks). The tool-belt itself is basically a handy place for the user to keep tools and items that may be useful in achieving these other objectives/tasks. Given this usage pattern the best method of assessment for the "move-with-user" tools is to allow the user to undertake a range of tasks over a long period of time and assess how beneficial the tool-belt and associated components were. The challenge for this trial is to compress this long-term tool-belt usage experience into a short interactive trial system which can (within the limited time span of these user trials) effectively measure the components effectiveness in their particular functional applications.

### *The Task*

Unlike earlier trials this trial does not have a specific task that a user is required to complete. Essentially this is an experience oriented trial and involves the user experiencing the world with the availability of a tool-belt and sub components for a fixed period of time. The task or activity that is required is simply to move within the 3D world and utilize the facilities provided by the tool-belt and other "move-with-user" components such as the pouch (for storing and retrieving items), the Swiss army knife (for accessing applications and tools) and portals (for linking to other workspaces). Whilst moving in the world the tool-belt and its set of sub components move with the user and provide their interactive functionality. By indicating to the user (i.e. at the start of the trial) that the purpose of the trial is to assess the tool-belt and the other "move-with-user" components, the user is encouraged to try these various components and report his/her opinions at the conclusion of the interactive experience.

Given this type of trial there are no quantitative measures that can be utilized to assess the effectiveness of the tool-belt (or any of the other tools), instead the assessment is based purely on the qualitative questionnaire results provided by the user at the conclusion of the trial.

### *The Trial Itself*

The user experiences two separate trial systems. Essentially, these are both simple examples of virtual environments however the two worlds are different in style to provide the user a differing type of experience.

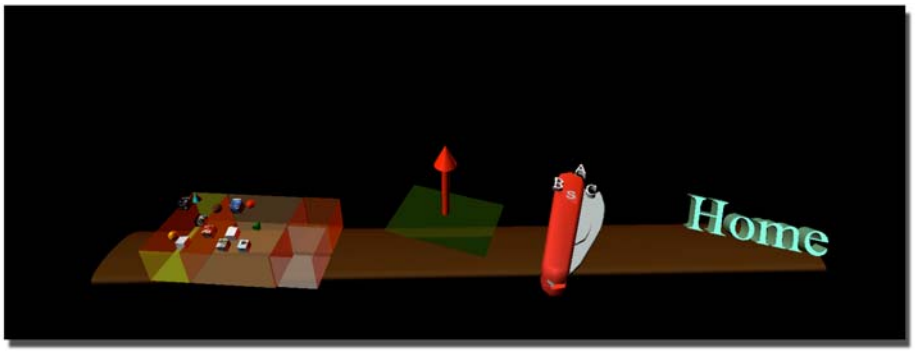



**Figure 7.38: Screenshot of the tool-belt used in all trials.**

In both systems the user has the same tool-belt (as shown above), which contains the following sub-components:

- The Pouch (leftmost on tool-belt above) - this pouch contains an existing set of items (intended to represent the type of items that a user would store in a pouch) spread amongst two of the four compartments.
- The Orientation Aid (next left above) - this simple “plane and normal” style of orientation aid proved the most effective in earlier trials and as such was adopted as the default.
- The Swiss Army Knife (next left above) - this knife has a subset of three tools (for simplicity represented as A, B and C).
- The Home Button (right in image above) - a simple button that returns the user to his start location should he wish to restart the trial.

The user carries this tool-belt through two worlds as a part of this trial. Those worlds are:

**The Empty Moving World** - this is a very simple world containing a single large terrain consisting of a set of gently rolling hills (as shown on the right).

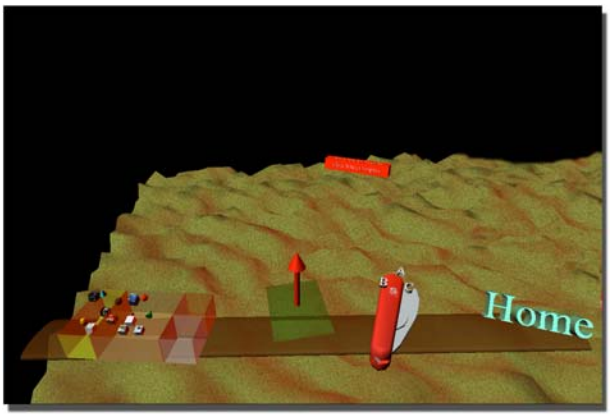



**Figure 7.39: The empty moving world.**

The world has few items in it, however it is quite active as the ground plane rotates, thus giving the user the experience of a world rotating underneath his/her location. Located in this world is a red rectangular prism which the user must locate and select to complete this trial. The intent with this world is to discover how useful the user finds the tool-belt when the world itself contains few items to interact with and the world has a motion of its own (i.e. separate to the users movement through the space).

**The Rich World** - unlike its predecessor this world contains many items, including portals past which the user moves carrying his/her tool-belt and interacting with the items in the space. This trial is time based and as such the user is able to move through and experience this space for a fixed period of time.

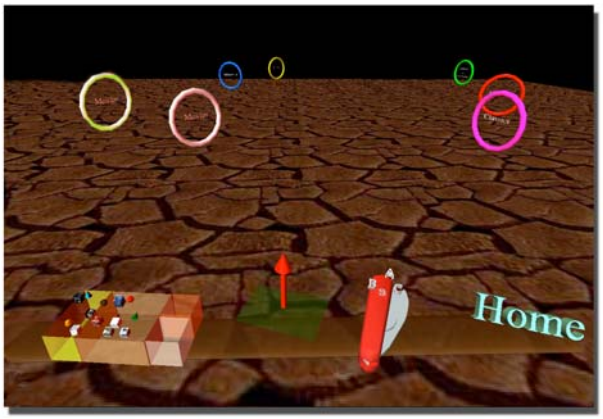



**Figure 7.40: The rich world.**

After experiencing each world the user is presented with a questionnaire (the results of which are more fully described in the following section), these questionnaires ask the user to assess the tool-belt in terms of how effective it was in providing both a sense of presence in the 3D space and also how useful the components were in providing their specialized functionality.

## Results for the Environment Interface

This trial was undertaken by six participants, with each participant completing the full set of tasks. The results from these user trials are summarized below, with results for each separate component described in the appropriate section. The tool-belt represents the overall group of components. Therefore the bulk of the questions presented to the users ask about the suitability and value of the tool-belt.

The tool-belt itself has several design concepts including the fact that it is largely made up of partially transparent items (this is to enable users to see-trough it into the world behind) and for these trials was permanently visible in the users view. The users answered questions (see below) relating to these design issues as part of the questionnaires at the conclusion of the interactive phase of the trials, the questions themselves and charts representing the answers to those questions are presented below.

Question: How much did the tool-belt aid your sense of presence in the world? (Not at all - Significantly)

Question: In providing a set of portable tools/items, how effective/suitable was this interface? (Extremely Unsuited - Very Effective)

Question: This tool-belt included a solid brown bar at the base, was this beneficial? (No benefit - Yes, beneficial)

Question: Parts of the tool-belt were partially transparent, is a partially transparent tool-belt better? (Much Worse - No different -Much Better)

Question: Do you feel you would find it easier to use now that you have experience with it? (No (it didn't improve) - Yes (easier in time))

Question: How much better/worse would these systems be if you could hide/show the tool-belt? (Much worse - Much better)

Question: Would you like to be able to resize/locate/personalize the tool-belt? (Not important - Yes, critical)

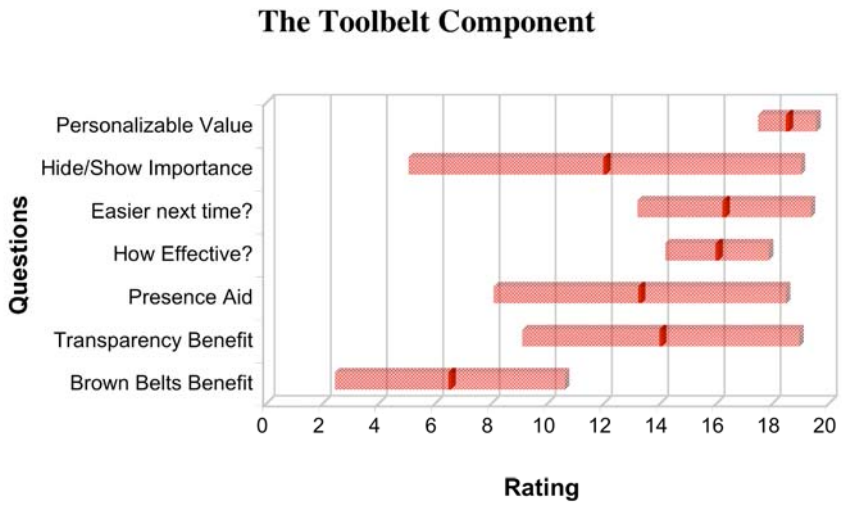



**Figure 7.41: Chart - The Tool-belt Component.**

As the chart above outlines the users generally found that the tool-belt was effective in its task, yet they strongly desired the ability to personalize its elements, appearance and size. Surprisingly the ability to hide/show the tool-belt was not a consistently desired feature (note the large range of values indicating that users are unsure whether this feature is desirable). As has proven to be the case with most systems the users again considered that this system would be easier to use with experience (although the range of user responses was broader here than in many other components). Clearly the physical “belt” element of the system was rated as being of little benefit to users and as such the tool-belt appears to be more effective simply presented as a series of tools without the need for the base belt feature. On the whole partial transparency appears to be beneficial, however the broad range of responses indicates that the transparency is not universally popular (perhaps allowing users to control the level of transparency may be a useful addition to this components capabilities).

These results indicate that the tool-belt is effective in its task, although there is clearly still potential to refine the features (and capabilities) that make up this component.

### *The Portal*

The portal component provides a mechanism to enable the user to move between different views/workspaces inside the 3D environment. During the trial the users were able to interact with “portals” and at the trials completion the participants answered the following questions about the portal components:

Question: In some of the systems the world contained "portal" items which would enable you to jump to another location (i.e. workspace in the 3D world). How valuable is it to be able to break a 3D world into different workspaces?

Question: How effective is the simple "portal" in indicating and achieving its task?

The chart on the right shows the users ratings of both the importance of such a workspace navigation feature and how effectively the portal component achieves this objective. This chart clearly demonstrates that almost all the users believe that a tool to navigate and manage multiple 3D workspaces is a critical component in a 3D interface. It also shows that the simple portal component used in this trial was effective (with 95% of users rating it above the halfway mark in terms of suitability) in this task although not necessarily performing to an ideal level. The fact that the users clearly left the upper (18-20) range of values (i.e. indicating that they felt there was potential for this task to be better handled) in the effectiveness scale indicates that although this component is effective it still has potential for improvement.

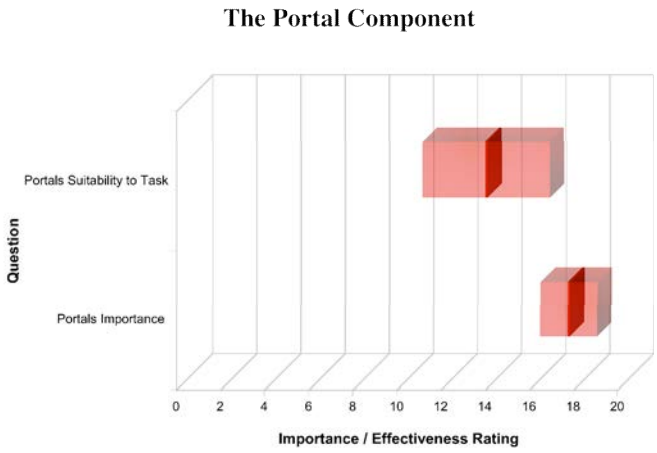



**Figure 7.42: Chart - The Portal Component.**

### *The Swiss Army Knife*

The Swiss army knife component provides the user with a portable device designed to hold a set of tools or applications. When activated this storage tool presents its child tools to the user for selection and use in the current task. Throughout the length of both trails the users were able to interact with and make use of the Swiss army knife and its features. At the conclusion of the trials the users answered the following questions about the Swiss army knife component:

Question: The Swiss army knife offers a tool container/selector, how effective is it in indicating and providing its function?

Question: The Swiss army knife provides the ability for you to take your tools with you through a 3D space, how important is this?

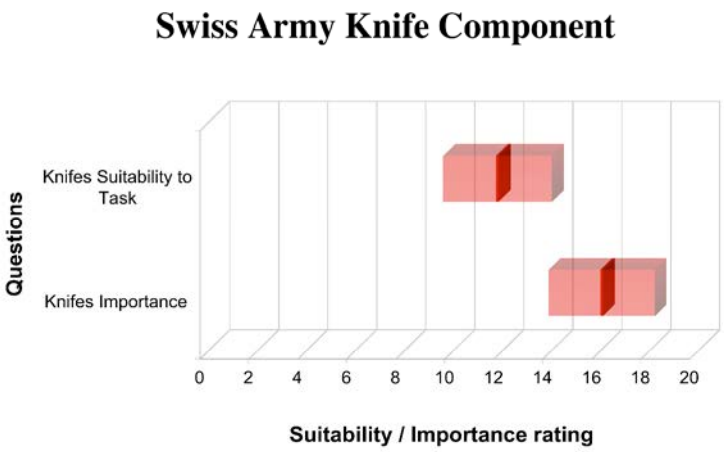



**Figure 7.43: Chart - The Swiss army knife component.**

As the chart on the left demonstrates the trial participants generally considered that the ability to carry a set of tools with them through space was a very important feature (as indicated by the average user response of 16 out of 20 on the importance scale). Given the importance of this feature the results for the Swiss army knife component indicate that it was not ineffective in its task, however the users did not find it highly effective either (i.e. average response of 12 out of 20 in rating its suitability). The user's comments provide some additional insight into the components key limitations by indicating that the knifes presentation of its options (i.e. in the form of a spinning set of items that enlarges when moved over) was effective, however it failed to indicate its purpose to users. That is to say that most users did not make the association between the "Swiss Army Knife" and a tool for holding other tools. The users appeared to prefer the Pouch as a mechanism for storing and retrieving items and considered the knife to simply be a less effective version of this concept. Clearly the objective of a "self-disclosing" component did not succeed for this component, however the fact that the users liked the presentation of the items indicates that there is potential (perhaps utilizing a different physical storage shape) for this component to be effective in its task.

### *The Pouch*

The Pouch component is designed to provide a simple storage location with which the user can place and retrieve items. It is intended to be somewhat like a briefcase/sports bag or general container. The pouch provides a set of compartments or sections into which these items can be placed. This "compartment based" approach enables the user to give some structure to how he/she stores these items. Essentially the pouch provides users with the ability to take items with them (and also to collect new items) as they travel through a 3D space. During both of the trials the users were able to utilize the features of the pouch and at the trials end they were asked the following questions regarding its effectiveness and value.

Question: The Pouch provides the ability for you to take items with you (and also to collect new items) as you travel through a 3D space, How important is this capability?

Question: The Pouch provides a simple storage, display and retrieval mechanism how effective is this in its task?

In responding to these questions the users indicated that the need for a storage tool that moves with them through space was critical and that the pouch component was very effective in providing this mechanism. As the chart on the right demonstrates, the users responses were all very positive and surprisingly were all within a tight range indicating that the pouch was both very effective and that almost all of the users rated it highly.

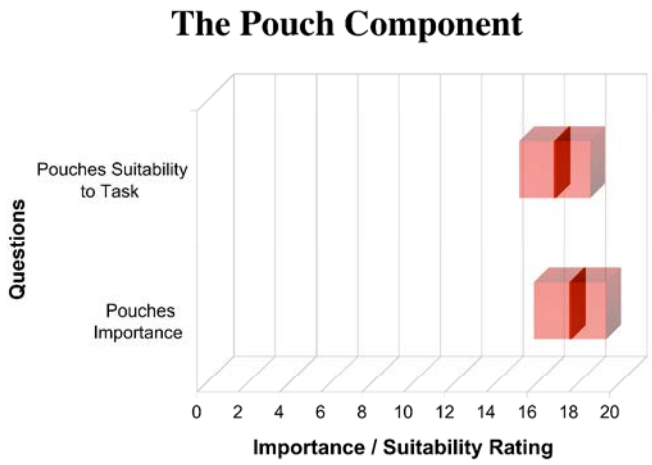



**Figure 7.44: Chart - The Pouch Component**

The pouch used in the trial contained a fixed (i.e. pre-defined) layout of compartments, in the real (i.e. the re-usable component) version these compartments can be shaped and coloured to suit the users personal preferences and as such the pouch will potentially be even more effective when used in this personalized form.

Each of the components outlined above provides a mechanism to support common tasks that users clearly require in an effective 3D interface. It is clear from these results that tools such as the pouch can be utilized to improve the users experience when navigating and working in a 3D environment. The final survey questions that were presented to the trial participants asked them to assess how valuable and enjoyable the tool-belt (i.e. the full set of components) was in providing a set of tool/items that move with a user through 3D space.

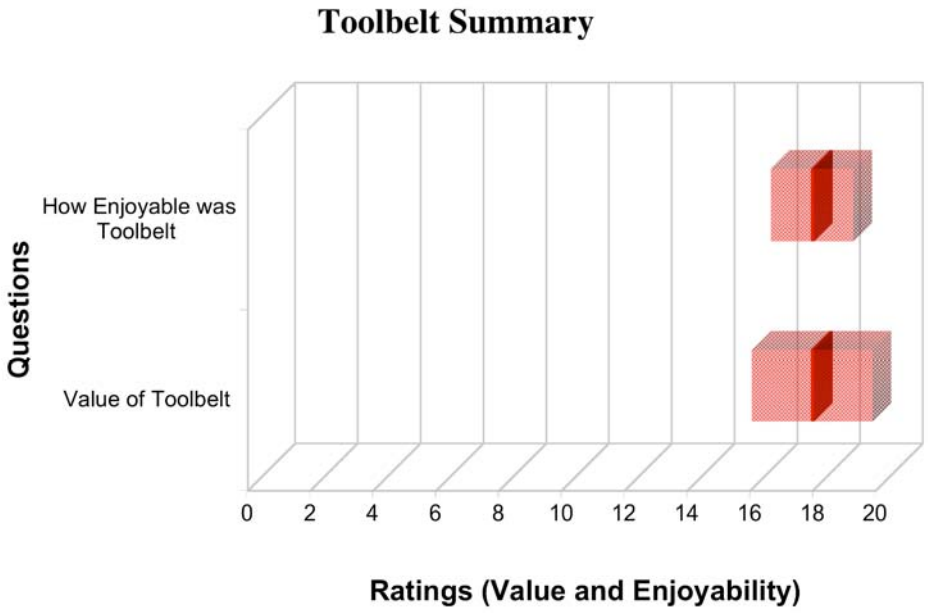



**Figure 7.45: Chart - The value of the tool-belt.**

As the previous chart shows the users found the tool-belt to be both very valuable and very enjoyable to use. Although some of the components were not ideal in their individual tasks, the overall system clearly achieved its desired goal of providing users with a set of tools that move with them through space.

## Analysis & Discussion

One of the key issues in assessing the value of these components involves determining to what extent the interactive trial results can be applied to a real world situations. Clearly moving around an environment and manipulating items within a pouch (for a short trial period only) is not the same activity as utilizing a pouch for storage and retrieval actions in a real task, or series of tasks. However it is a very similar task and does provide an insight into the users thoughts and functional requirements. From the results it is evident that the tools presented here are effective in providing the basic set of capabilities that the users desired. Of particular interest is the fact that the users had a strong preference for the ability to personalize the elements and the tool-belt itself. The components such as the pouch already provide this capability, however it is clear that enabling the user to resize, reposition, set transparency and generally “personalize” their tool-belt is an important feature to be added to this component in the future.

## Environment Interface Results Summary

The results from these trials clearly identified the fact that users would like the ability to carry tools and other items with them as they move throughout three dimensional spaces. The components developed in this research achieved generally high results in terms of their suitability to these tasks. Each of these components has contributed to the understanding of “move-with-user” tools, with some of the components developed here, such as the pouch clearly being very effective and popular with users in their tasks and others (such as the Swiss army knife) being functional yet not ideally suited to their challenges. The success of the pouch component in particular identifies that a simple storage tool that enables users to collect and retrieve items from a simple yet structured component is both beneficial and can be implemented in a functional and re-usable form.

The overall concept of a set of “move-with-user” tools is clearly an area of great importance for future 3D interfaces, with potential applications in areas such as virtual and augmented reality as well as more mainstream 3D user interfaces.

## Assessing the Circulatory System Interface

The circulatory system interface is the largest and most complex 3D interface component developed in this research. Its primary objective is to provide a 3D system for presenting large structured sets of data to a user. Utilizing an active presentation system the circulatory system represents a new type of interactive component. Its ability to function in its specified task is measured in the user trials described below. The task of selection from a large structured set can be handled using a variety of interface components, for the purposes of this trial the user undertakes selections using the two key methods (i.e. each described more fully in their earlier sections, the Window-Box based system (as described in the section *The Simpler Components - The Window-Box)* and the Circulatory System (as described in the *Circulatory System*). The performance results and user opinions of these systems provide the core of the data for analysis.

### *The Task*

The basic task that the user is required to complete involves locating a particular movie item from within a structured movie set/library. This library contains 112 unique movies each represented in the form of a cube with the specific movies image/poster mapped to its side (as shown on the right). These movies are broken into a set of categories and sub categories based on the genre or style of the film. This categorization is based on the following structure where each movie is categorized into one of the 5 top level categories and then potentially sub-categorized within that into one of the 2nd level categories.

**Figure 7.46 A movie item.**

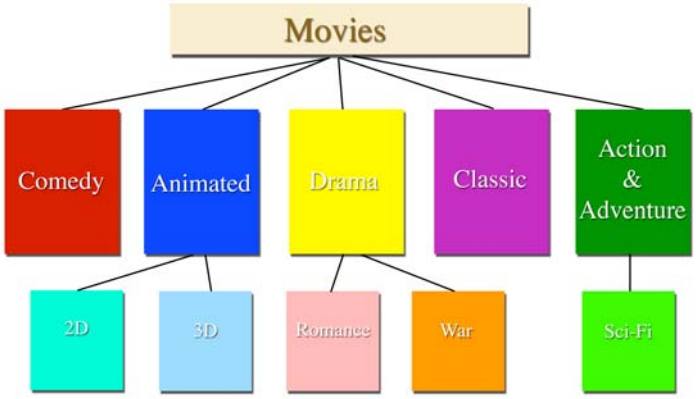



**Figure 7.47: The movie library layout.**

The selection task that the user must undertake involves searching through this structure and locating specific items. To test differing types of search activities the user undertakes two selection tasks with each interface style. The first is the “targeted task”. This task involves the user locating a specific known item from the set. This item is known to the user and as such he/she will benefit from the logical break-up/structure of the set of data when seeking it out (i.e. will recognize the movie and hence quickly move to the category it belongs in). The second task is the “browsing task”. This task requires the user to find a

particular item which he/she does not recognize (it is actually not a real movie and is simply a “Find Me” text word inserted randomly into the library) in the overall set. In this task the user will need to browse through the movie library seeking his/her target.

During all of these trials, data including the time taken, the number of incorrect and the number of correct selections is recorded. In addition the users questionnaire responses are recorded. Given the different types of tasks, the data recorded will be of differing relevance in different trials. For example, in the browsing task, time in particular is less important than the users rating of how useful the presentation style is; for the targeted task, time is critical. Taking these factors into account all the data is recorded for all systems and that data is then analysed based on its relevance for the particular task.

### *The Trial Systems*

There are two systems being tested in this set of user trials. They represent two quite different methods for presenting structured data. The Window-Boxes interface is essentially a 3D version of the widely used desktop and window interface used in 2D GUI systems and the Circulatory system represents the new 3D interface developed in this research. Both systems are tested in both the targeted task and the browsing task and the results are presented in the section below. A brief description of the two interactive systems utilized for the tests follows:

**Window-Boxes**


**Figure 7.48: The Window-Boxes interface.**

This interface stores the data in a window based categorized storage system, where each category has a Window-Box containing the movies that fit that category (eg. the coloured

boxes above are each examples of Window-Boxes). These Window-Boxes can be opened by selecting them in their parent Window-Box. They can also be closed by clicking the ‘X” in their upper right corner. To navigate through this structure the user must actively seek out each Window-Box and check it for the desired target.

**Circulatory System**

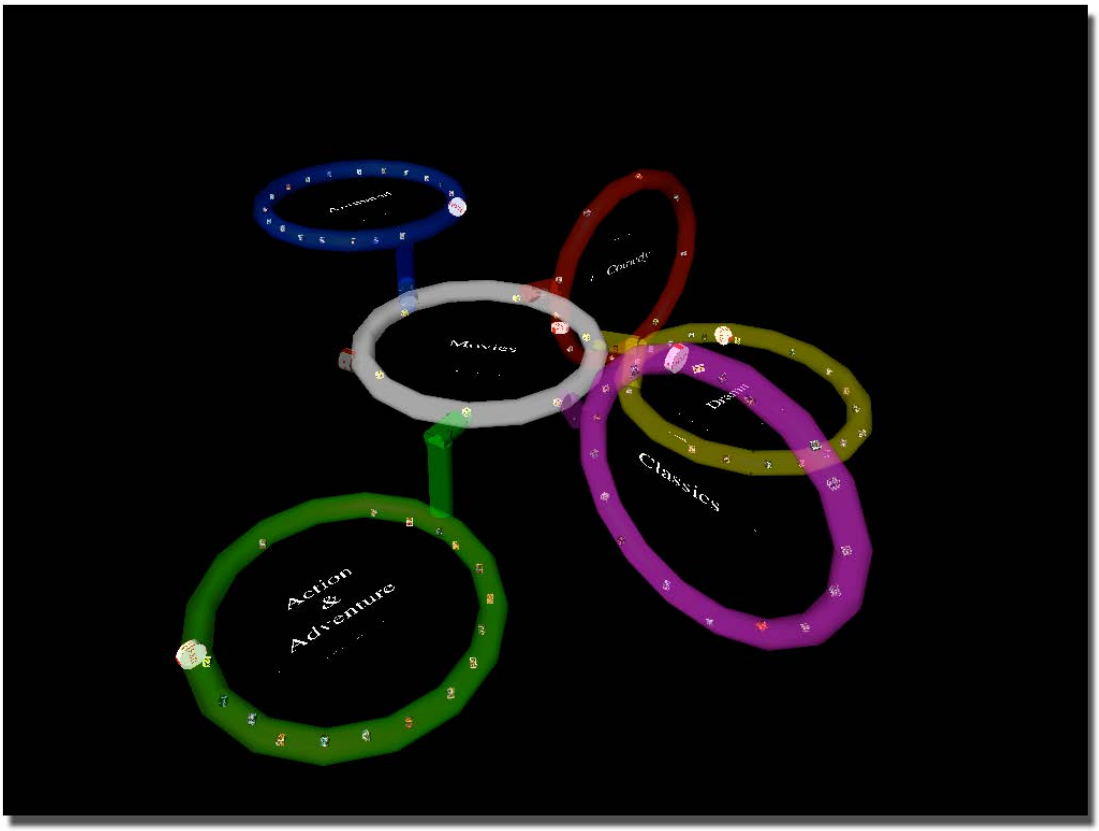



**Figure 7.49: The Circulatory System interface.**

This interface uses coloured rings of items to represent categories (as shown above), the user is able to click these categories to go inside the rotating tube and have the items flow past his/her location. One of the key differences presented in this new style of interface is the fact that it actively presents its data to the user. Once inside, the information flows in loops past the users location. By default the user takes all the branches and consequently by simply entering the circulatory system at any location the user can simply watch as the entire system cycles by. This active form of interface is very different from the static form of the tree of Window-Boxes above and the trial results indicate which is most effective for the various tasks.

### *The Trial Itself*

The trial itself is broken into two separate phases, the first is primarily an assessment of the 3D Ring component (i.e. a sub-ring of the full circulatory system). In this phase the user is presented with three different types of ring presentation and asked to assess which is best for its task. The three ring types are:

Basic Ring: The basic ring simply involves a ring with individual items spaced evenly through the ring. It has no surrounding structure and at each location in the ring there is only a single item.

Tube Ring: The tube ring involves a ring with individual items spaced evenly through the ring. It has a surrounding tube structure that provides a shape within which the items move. This tube also provides a sense of the ring that is currently active (i.e. user is inside a blue ring which is the X category). As with the basic ring at each location in the ring there is only a single item.

Ring of Rings: The “ring of rings” involves a ring with “sub ring items” spaced evenly throughout the base ring. It has no surrounding structure but at each location in the ring there is a full “sub-ring” of 8 items (note that this ring is rotated so that the user will look through the centre of the “sub-ring”). Clearly this system is representing 8 times the number of options and therefore is expected to be more complex for users.

Having selected the target from within each of these ring types the user provides questionnaire responses to specify which system he/she found most/least effective, thus completing the first phase of this trial.

The second phase involves using the full “Tree of Window-Boxes” and “Circulatory System” to locate the target items. In itself this part has two sub parts, those being the “targeted” and “browsing” activities. At the completion of these tests the user answers a series of summary questions to finish the trial.

## Results for the Circulatory System Interface

This trial was undertaken by nine participants, with each participant completing the full set of tasks. The charts that follow indicate the order in which the trial systems were presented to the user, with the first system at the base of each chart’s results. The results from these trials are presented in two separate categories. First are the results from the “Ring Component” phase of the trial presented in full include the summary results for these elements. Then the second phase results are presented, these incorporate the comparative results for the larger Circulatory system and the Window-Boxes equivalent.

### *Part A: Results for the Ring*

In assessing how effective the various forms of ring interface are in their task of presenting a set to the user, the results for the various systems were as follows:

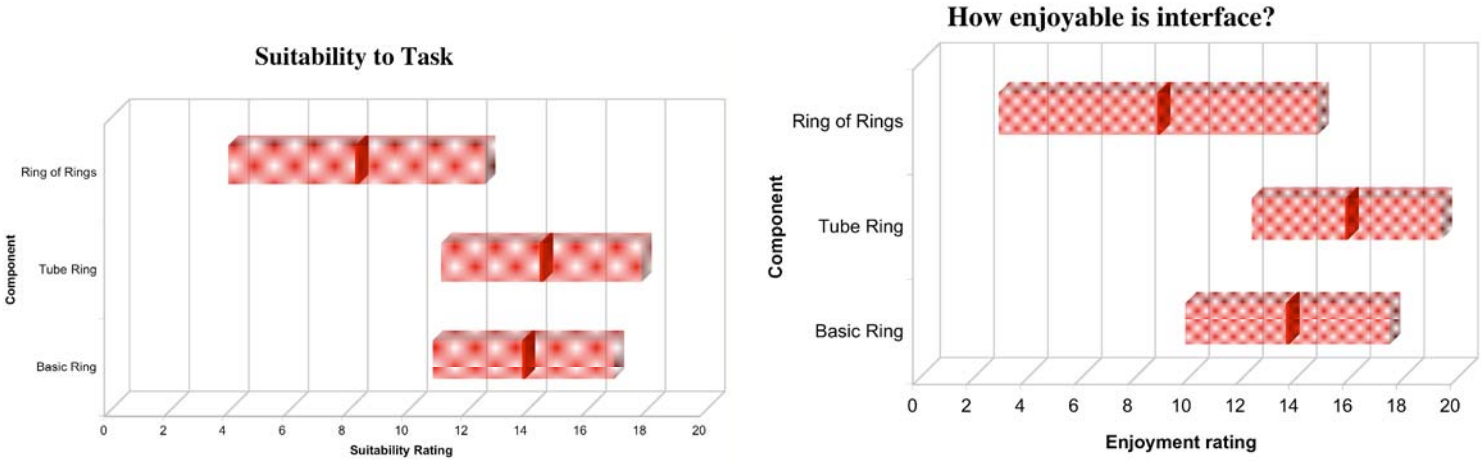



**Figure 7.50: Chart - Suitability of Ring Interfaces.**

As the charts indicate the Ring of Rings was clearly both less suitable and less enjoyable to use in completing this task. Interestingly the Tube-Ring rated as both the most effective and the most enjoyable (although only slightly above the Basic-Ring). Given that the quantitative measures of time taken and error rates were identical for all systems this indicates that the users found the nature of the surrounding tube helpful.

The apparent failure of the ring of rings system is most likely due to the density of information. Many users commented that they found the “ring of rings” had too many items which made it complex. However had it had a smaller number per location (i.e. there were 8 items per location) they thought it would be useful.

At the conclusion of the test the users were asked to identify which system they preferred for this type of task. Interestingly 60% chose the Basic-Ring while the other 40% selected the Tube-Ring (i.e. 0% chose the “Ring of Rings”). Given that the earlier charts indicated that the “tube ring” was more effective, this was a surprising result. The general comments indicated that the users liked the simplicity of the basic ring. In addition several users disliked the fact that the tube blocked them from clicking items until they came around the corner. Overall it is clear that the Basic-Ring and Tube-Ring are similar in terms of effectiveness with both offering differing features, perhaps allowing the user to choose to either “show” or “hide” the surrounding tube would be the best option.

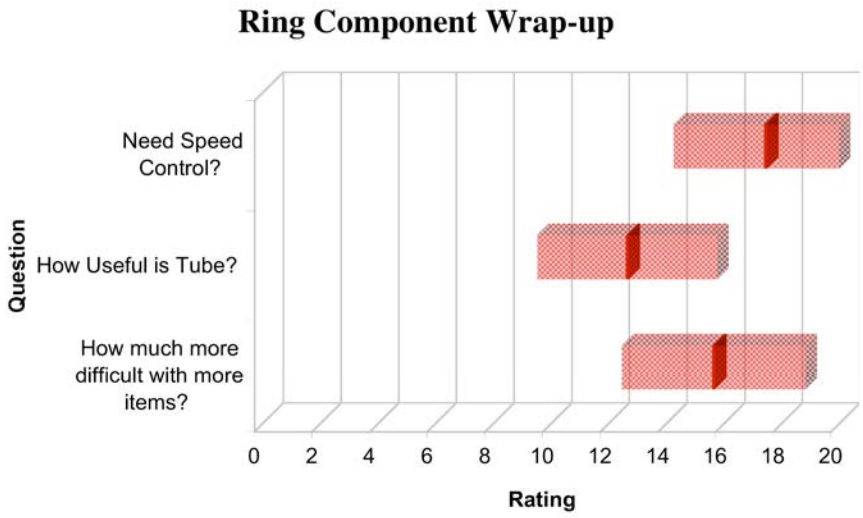



**Figure 7.51: Chart - Ring component wrap-up**

The wrap up questionnaire provided some additional information including the fact that there is a clear need for the ability to control the flows speed (note that the normal version of the ring incorporates such a control). The fact that more items makes the challenge harder was already identified by the "ring of rings" system however this result clearly focuses on the fact that the users found the density of 8 items per sub ring to be too much (several users recommended 3-4 items as an ideal amount).

These comparative results, when viewed as a complete set indicate that the users found both the Basic-Ring and Tube-Ring to be effective, although they had a slight preference for the Basic-Ring. They liked the idea of the Ring of Rings but found the amount of information it presented overwhelming.

### *Part B: The Circulatory System*

Results is this section of the users trials represent direct comparisons between the newly created, three dimensional, active presentation system of the "Circulatory System" and the "Window-Boxes" interface which is fundamentally designed along similar lines to those followed by the widely used 2D GUI window systems.

The performance figures shown below are specific to the particular task (i.e. either the targeted search or the browsing activity) that the user is undertaking. The first set, presented below, relate to the targeted task, where the user knows the target he is seeking and is attempting to get to it as quickly as possible.

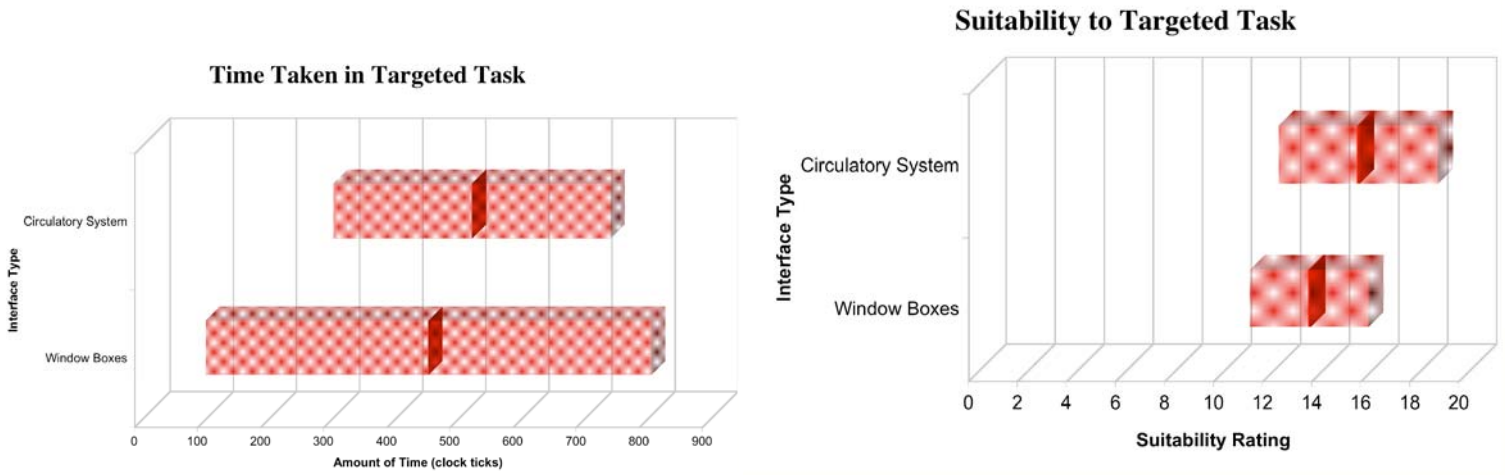



**Figure 7.52: Chart - Effectiveness of structured set systems.**

As these charts show users took similar lengths of time with both systems to complete the task and the users rated the circulatory system as being slightly more suited to the task. Given that this targeted searching activity is not the primary objective for the circulatory system this is an unexpected positive result.

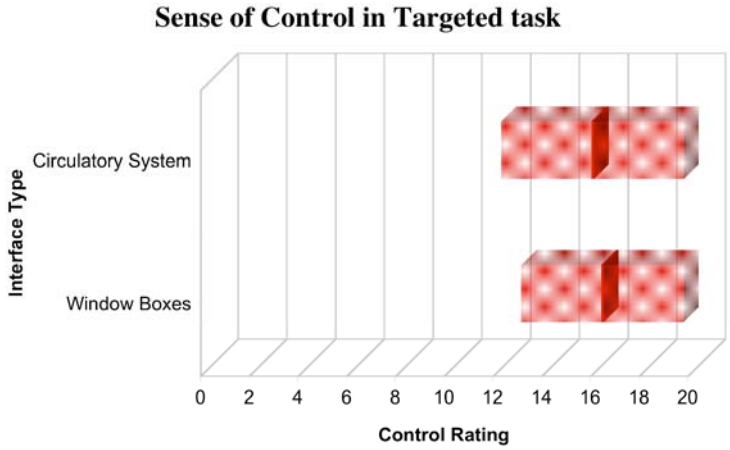



**Figure 7.53: Chart - Control in the structured set systems.**

With the circulatory system presenting a very active form of interface, there is a clear risk that the user will lose the critical sense of controlling his/her environment. The chart above indicates that the circulatory system was effective in maintaining this sense of control, to the point where users found they had similar levels of control in both the active circulatory system and the static windowing system. While this chart shows that the users felt in control of both systems the following chart clearly shows that the users found the circulatory system to be more enjoyable to use. This may be due to the active nature of the system (i.e. making the selection less of a navigating task and more of a viewing experience).

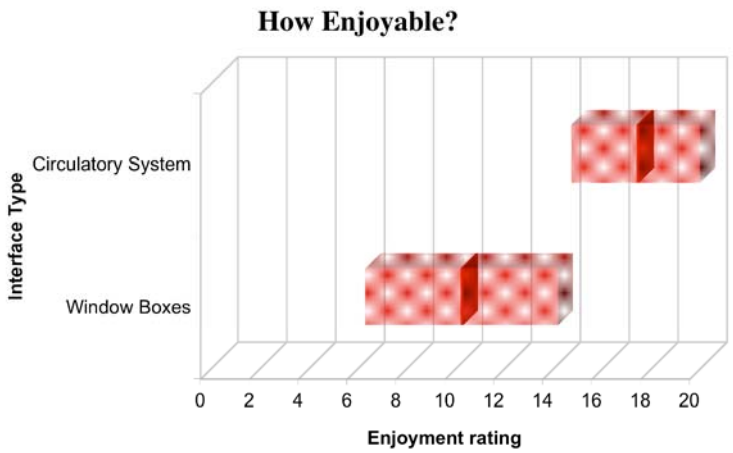



**Figure 7.54: Chart - How enjoyable are structured set systems?**

The targeted search task undertaken here was not the primary focus for the circulatory system, in fact in real world use it is unlikely that many users would use a system like either of those described here when searching for a specific known item. The more likely scenario would see the user talking advantage of an automated search tool (i.e. search engine) to seek the particular target item and bring that item directly to the user. The activity of manually searching through a structured data set when seeking a known target is likely to only occur rarely. However, that said, the circulatory system created in this research performed surprising well in this task, clearly performing as well or better than the widely used window based system in terms of both functional effectiveness and user preference.

The second type of application that is envisaged for these tools for presenting large structured sets is in the browsing situation (i.e. the task is to find an unknown item). In this situation the user wants "browse" what is available and make a selection from what he/she sees. For this critical browsing task (i.e. browsing is generally the method most people use for numerous tasks on a daily basis in the real world) the user results were as follows.

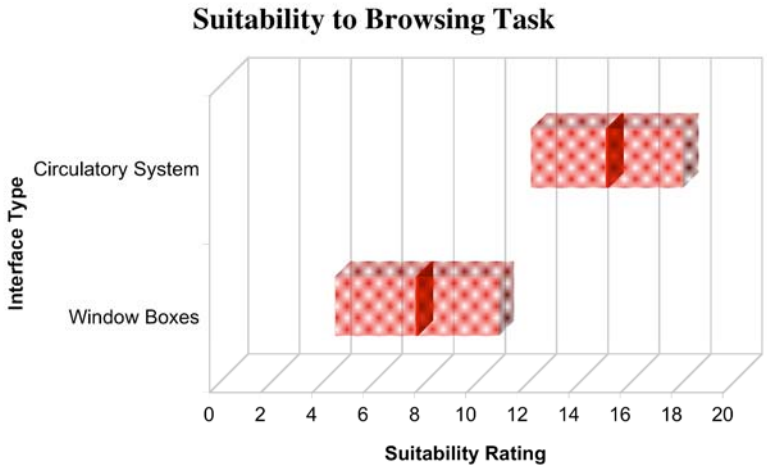



**Figure 7.55: Chart - Browsing task interface suitability.**

As is clear from the suitability chart on the preceding page users found the circulatory system to be significantly more suited to this browsing activity than the window box system. On the critical issue of retaining the users sense of controlling the system, once again (as was the case for the targeted task) both systems produced similar results (see Figure 7.56), indicating that the users felt in control of both although the large ranges indicate that both systems had strengths and weaknesses in maintaining the users sense of control.

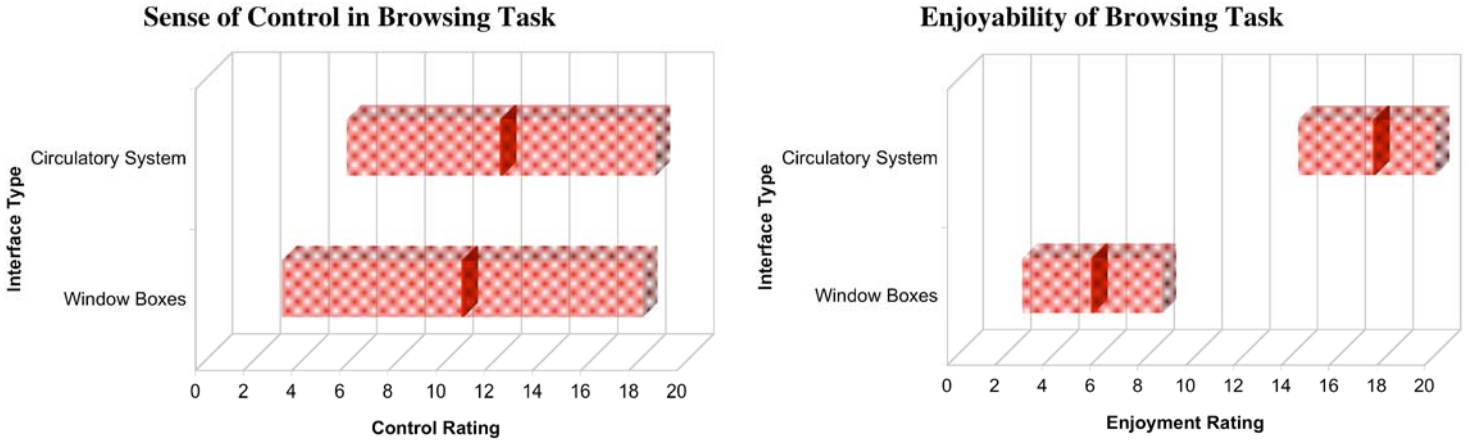



**Figure 7.56: Charts - User control & enjoyment.**

In terms of enjoyability, once again the circulatory system was rated as significantly more enjoyable to use. This may be partially due to the "new" nature of the interface, however it is clear that the flowing system is preferred by most users when undertaking a browsing task of this kind.

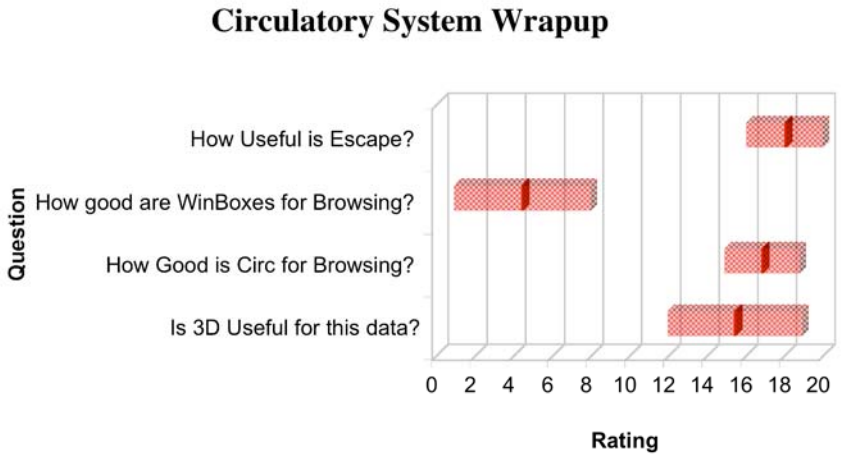



**Figure 7.57: Chart - Wrapup for structured set selectors.**

As the wrap-up chart above, which shows the users responses to a set of final questions, demonstrates most users wanted the ability to "escape" from the current ring. Although

this did not present a problem in this trial users envisaged a situation where they could enter a ring by mistake and would have to wait for the exit to come around. This would not be desirable and hence they indicated that an escape mechanism in the circulatory system would be very beneficial. Such an escape exists in the normal version of the circulatory system (the basic version that incorporates a flow control mechanism, one element of which is a tool to step out of a level) however for this trial those controls were removed in order to maintain consistent rates of flow between users. Given the importance that the users place on this escape mechanism perhaps the tool for this task should be made into a more notable item in the flow controls.

The results of the users final direct assessment of the two systems capabilities for the browsing activity clearly demonstrates the fact that the circulatory system was preferred by users for browsing. This result was also reflected in the users choice of preferred system for the browsing task where 100% selected the circulatory system as their preferred browsing option. In the targeted task the preferred options were split with 50% choosing Window-Boxes and 50% choosing the circulatory system. The results for the final question “Is 3D useful for presenting this type of data?” indicated that most users found 3D to be very useful for this type of task.

## Analysis & Discussion

It is clear from these results that the flowing system is effective for this type of browsing task (ie. using small to medium sized selection sets). However what of the possibility/potential of developing flowing window boxes or another form of flowing interface. Is it simply the flowing concept that makes this interface work or is it the combination of three dimensional space and a flow that makes it work? Results from the earlier flow component showed that flows moving in 2D space (i.e. the LefttoRight and ToptoBase systems) were significantly less effective than their 3D depth oriented equivalents. This indicates that, in the unstructured data case, it is not simply the flow but the combination of flow and depth that makes the system effective. These trials have shown that the nature of the 3D flow works very well not just for unstructured data but also for structured data. Given this data, and the similarity in the nature of presentation it is unlikely that a 2D flow oriented interface would be as effective as the 3D circulatory system described. In fact it is most likely to be significantly less effective as was the case for the simple 2D flow components.

## Circulatory System Results Summary

The results summarized above clearly show that the trial participants found the circulatory system created in this research was more effective and more enjoyable for the task of selecting items from large structured sets than the comparative interfaces. Both systems provided the users with similar levels of control however the users found the active presentation of the flowing set of data (i.e. in the circulatory system) to be a more effective means of viewing this data for both the specific targeted search task and the browsing task. The differences in terms of results were more pronounced in the browsing task where the circulatory systems success was more notable and represented a significant advantage over its counterpart.

The results shown here have clearly demonstrated the benefits of a flowing set of data in a 3D space, particularly when applied to large data sets (either structured or unstructured). All of the flowing interfaces designed in this research (i.e. the Flow, Tunnel and Circulatory system) have shown enormous potential and have clearly been more successful than competing systems in presenting these large sets of data. Given the number of real world applications in which these types of interface components can be applied (i.e. from online stores and libraries to search engines) these discoveries represent a significant step forward in the development of effective 3D user interfaces and the identification of tasks and activities in which these interfaces can most effectively be utilized.

### *7.2.2 The 3D Interface Builder*

As described in the *3D Interface Construction Tool* chapter earlier in this thesis, the 3D interface builder application is designed to provide its users with the tools to quickly and easily construct interactive 3D user interfaces. This section looks at assessing the builder tools capability to meet these objectives particularly in the areas of:

- Enabling non-programmers to develop interactive 3D user interfaces.
- Providing a faster, easier and more reliable method for developing 3D interfaces.

#### Assessing the Builder Tool

Knowing that the primary function of the builder tool is to enable the development of 3D interfaces. The best way to assess its capabilities is to use it in exactly that task. Unfortunately this interface construction task is a difficult and challenging activity for even the most skilled developers. The initial trial that was implemented essentially involved setting the users the task of constructing a simple interactive 3D interface using several different construction methods, including the builder tool, hand writing using VRML and others. Unfortunately the first result that this trial identified was the fact that virtually none of the trial participants had the necessary skills to hand write the VRML interface (let alone consider the more complex programming / OpenGL options), thus giving no means of comparing the various systems in terms of development speed, development errors and general usability. In itself this is an interesting result as it clearly identifies the fact that the currently available development tools are not capable of providing the necessary functionality to enable most users/developers to construct 3D interfaces. However, due to these circumstances, the results relating to development using the builder tool, as directly compared to other systems (eg. quantitative assessments), are based on other measurable factors.

Given this limitation the interactive user trial was altered to focus on assessing the builder tools ability to provide these “unskilled developers” with a 3D interface development platform. Hence the trial in its final form involved measuring the users 3D interface development capabilities both with and without the aid of the builder tool. In doing so identifying if the tool itself is useful in empowering these potential 3D developers. To implement this trial involved initially obtaining a measure of the users development skills (i.e. without access to the builder tool), this was achieved by asking the user a series of questions (as outlined in the following section). The answers to these questions provide a base level for understanding both the users initial skill level and the users level of confidence in developing 3D interfaces.

Having completed this stage the users were then provided with a single page of information (see *Appendix B: 3D Interface Development Information*). This information describes the task (i.e. the interface that needed to be implemented) and also provides a basic introduction to the builder tool’s capabilities. The user then begins using the builder tool and developing the interface as specified. During this development phase the main quantitative data recorded involves the time taken to construct the final system. At completion the final system is then compared to the target to provide an additional set of

quantitative results (i.e. giving quantitative measures of how correct (when compared to target) the resulting interface is, and how long it took to develop). In the final stage of this trial the user is presented with another questionnaire, this time asking similar questions to those from the first questionnaire. These questions are designed to determine if the user (now having experience with the builder tool) is capable of developing 3D interfaces and to what level the user has confidence in these skills.

Essentially this whole trial involves testing whether a user, who at the beginning of the trial has a certain skill level/capability, has a greater skill level/capacity to develop 3D systems through the use/availability of the builder tool and the new development facilities it provides.

## Results for the Builder Tool

As the development task described above is a more involved and time consuming activity than those undertaken in the other trials, this trial was carried out on a specific, case study based, smaller group of trial participants. The results presented here represent a set of three case studies, each identifying a different level of user skill and the impact that the builder tool had on their development capabilities and confidence.

### *Case Study A: The Experienced 3D User*

This user fits the description of being an experienced software developer with strong experience using computer systems in general, including 3D software and systems. This user identified their skill/experience levels prior to this task as being:

- A regular user of computer systems
- A regular user of 3D systems

In addition this user had high levels of software and interface development experience (although limited experience in 3D user interfaces). The chart below shows the users initial responses (i.e. before undertaking any development) to these skill based questions.

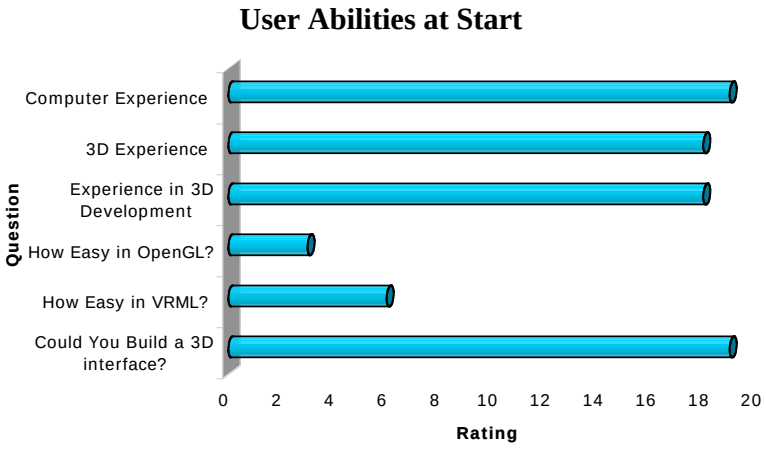



**Figure 7.58: Chart - Experienced user initial skills.**

The experienced user had little difficulty building the specified interface and completed the task in 4 minutes and 22 seconds. This was a surprisingly quick performance and was interesting in that this same trial had taken the author 4 minutes to complete when testing it for use. This fact identifies that the conceptual side of the task was easily understood by the experienced developer as was the method through which the tool provided an implementation. In this case the development time taken simply reflected the actual process of setting the values within the application itself rather than time spent understanding the workings of the builder.

The resulting system generated by the experienced user was functionally correct and essentially identical to the target system described. At the conclusion of the development phase the experienced user answered the following questions.

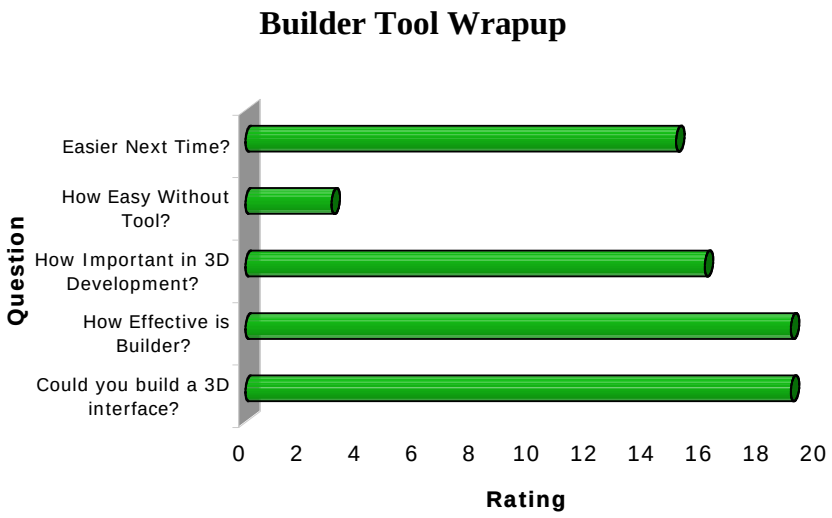



**Figure 7.59: The experienced users post development questions.**

The experienced user started the trial with a high level of confidence in his ability to construct a 3D user interface. The trial itself did not alter this users confidence in this area. The user did however rate the builder tool as extremely effective in its task as a 3D interface development platform and also indicated that should he build a large complex 3D interface he would use the builder tool to create it.

In summary this user had the skills to construct a 3D interface using a number of methods and found the builder tool highly effective and easy to use in this task. Indicating that for future projects he would utilize the builder tool as his development option of first choice.

### *Case Study B: The Mid-Level 3D User*

This user fits the description of being an experienced computer user with some experience in 3D systems but little or no software development skills or experience. This user identified their skill/experience levels prior to this task as being:

- A regular user of computer systems
- A user of 3D systems

This user represents the portion of the broader computer user population who are likely to have an interest in 3D development tools such as the builder tool developed in the research. That is a computer user who is interested in 3D graphics but does not have the necessary programming skills to be involved in 3D interface development. The chart below shows the users initial responses (i.e. before undertaking any development) to these skill based questions.

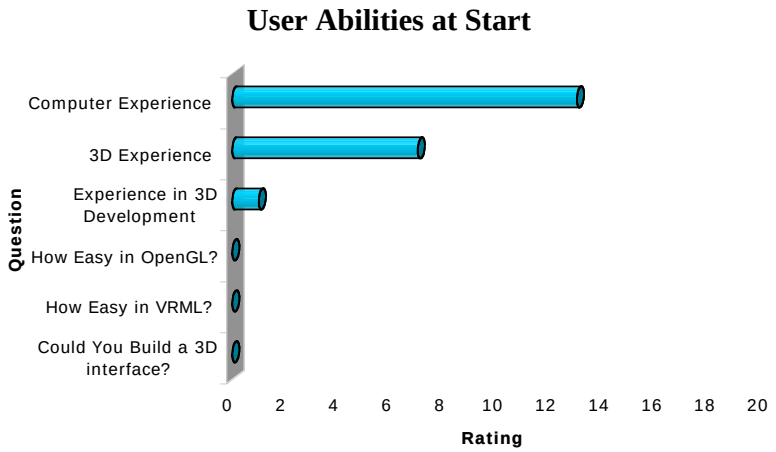



**Figure 7.60: Chart - Mid level user skills pre trial.**

The mid-level user successfully completed the development of the target system in 18 minutes and 32 seconds. Although this is significantly longer than the time for the experienced developer, the mid level users resulting system was equally correct in functional terms.

During the development process this user raised the issue of confusion between “Proximity” sensing and “isOver” sensing, indicating that the difference between the two was unclear. This problem stems from the fact that in the 3D world there is a sense of both the user location (i.e. where the users view is) and the users pointer (i.e. where the user is interacting). This makes for the capability of items that are sensitive to both pointer interaction (click, drag and isOver) and user interaction (move within range, visibility etc). Therefore an understanding of these two interactive systems is beneficial in aiding developers.

Aside from this the mid-level user found the development process to be very effective and at the conclusion of the development phase provided the following results.

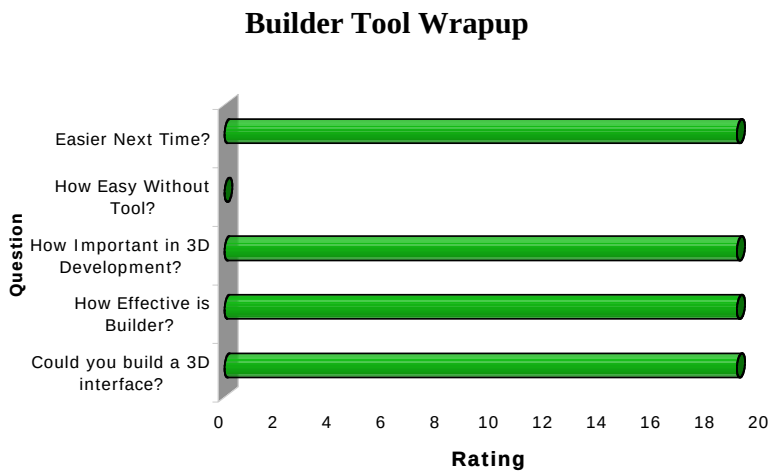



**Figure 7.61: Chart - The mid-level user post development.**

The mid level user represents one of the key targets for the builder tool. They started the trial with no confidence in their ability to build a 3D user interface. However after working with the builder tool to complete the trial task they indicated a high level of confidence in their ability to construct a 3D user interface. This result clearly shows the power that the builder tool places in the hands of a mid-level user, taking them from a developer with no ability to create 3D interfaces to an empowered developer who is confident of their ability to build complex 3D interfaces. The other results shown above indicate that this user found the tool to be very effective as an interface construction tool and that without it they would not be able to construct anything.

In summary this is a very pleasing result and demonstrates that users with some 3D experience but limited development skills can use a high level builder tool to develop 3D user interfaces. Hence the high level builder tool proposed and constructed in this research has contributed to the field of 3D user interfaces by empowering a large body of mid level users to enter the 3D interface development field.

### *Case Study C: The Inexperienced User*

This user fits the description of being an experienced computer user with essentially no experience with 3D systems and definitely no software development experience or skills. This user identified their skill/experience levels prior to this task as being:

- A regular user of computer systems
- A user who rarely (or never) uses 3D systems

This user represents the broader population of computer users who have experience in using computer systems but have not been involved in 3D graphics nor have they been involved in any software development. The chart that follows shows the users initial responses (i.e. before undertaking any development) to these skill based questions.

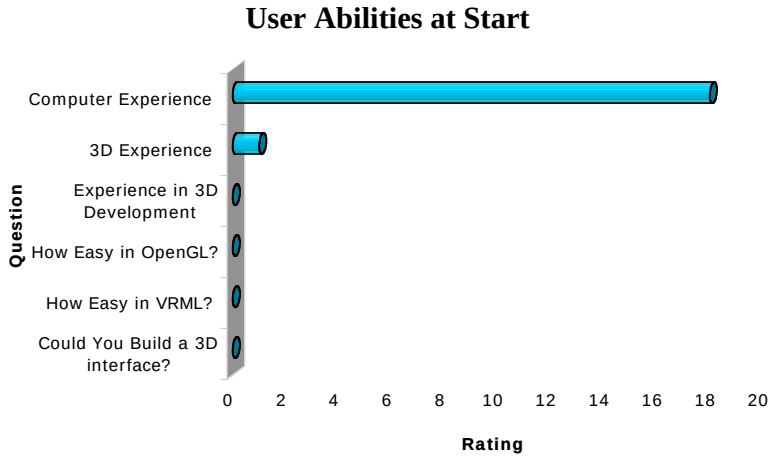



**Figure 7.62: Chart - Inexperienced user skills pre-trial**

Surprisingly the inexperienced user took slightly less time than the mid level user to complete the target system. Taking a total of 18 minutes and 18 seconds to correctly implement the target interface. Although this is slightly faster than the mid level user this difference is minor and based on observations of both users it is most likely due to the fact that the inexperienced user was simply trying to complete the task as quickly as possible where the mid-level user was attempting to understand the tool for future use.

During the development phase the inexperienced user indicated that the number of actions (i.e. interactive actions to take when a sensitive item is activated) available is very large and the number was confusing. As a result perhaps some structure in presenting these “action options” may make it more effective (eg. categorize into actions that move items, actions that start programs…).

At the conclusion of the development phase the resulting system generated by the inexperienced user was functionally correct and error free. At this time the inexperienced user answered the following questions.

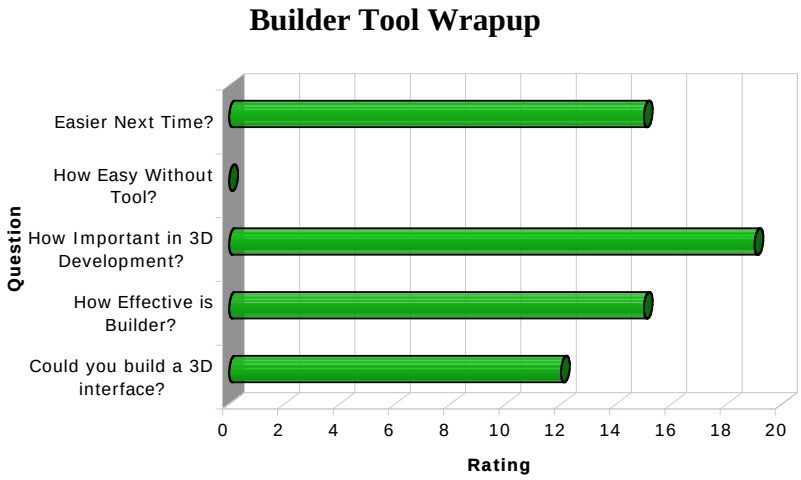



**Figure 7.63: Chart - Inexperienced user responses post trial**

The inexperienced user represents one of the users who is a target for the builder tool, however it was uncertain to what extent this group would be interested in 3D development and to what extent they would be capable of it. They started the trial with no confidence in their ability to build a 3D user interface (i.e. zero out of twenty on the scale). However after successfully completing the development task with the builder tool they indicated a higher level of confidence in their ability to construct a 3D user interface (i.e. 13 out of 20). This result clearly shows that inexperienced users, like the mid level users, were capable when provided with a builder tool of building 3D interfaces. The other results shown above indicate that this user found the tool to be effective as an interface construction tool (although not to the same level as the other case studies) and that without it they would not be able to construct 3D user interfaces.

In summary this is another very pleasing result and continues to demonstrate the fact that users without existing development skills can, when given access to the right tools, develop effective 3D user interfaces.

## Analysis & Discussion

The test interface, as required to be constructed by participants, was extremely simple, and there may be some question as to whether a simple challenge of this kind is a fair assessment of a users ability to develop a larger scale 3D interface. It is true that this system is small however it demonstrated the users ability to add interactive items to the space, locate them, give them appearance attributes as well as the key skills of attaching actions and interactivity. Although the overall interface constructed is simple, the tasks undertaken to build it cover the range of key implementation tasks needed for more complex systems. As such a user with the simple set of skills demonstrated in this trial could reuse these same actions to construct much larger and more impressive systems. There are certainly some specific tasks that this trial does not test. However, as a broad representation of the tasks involved in interface development this task is effective.

An interesting side note from this user testing is the observation of how the different users approached the task. The less experienced users relied heavily on the aid

paper/document for steps on how to achieve various tasks, always referring back to it when unsure. At the other extreme was the experienced user who jumped right in and simply began trying the facilities of the package. Both approaches produced effective results however the different approaches were very notable.

## 3D Construction Tool Results Summary

These results demonstrate that the conceptual idea, developed in this research, of a tool that enables the programming of a 3D interface without requiring the user to undertake or even understand programming, could bring development options to a new set of developers. Not only in the form of a concept but as a practical real world tool that has been implemented and in trials has demonstrated that, even in the hands of an inexperienced user, it was clearly able to create interactive 3D interfaces and give this user confidence in his/her ability to construct these systems.

The results from this trial clearly show that the concept of a high level 3D interface builder tool can enable users of all experience levels to become developers of effective 3D user interfaces. The innovative approach, developed in this research, to avoiding the need for the developer to program the actions associated with interactive 3D applications has empowered a new body of developers, enabling them to construct the type of systems that were previously well beyond their reach and therefore has significantly contributed to the current knowledge and future potential of 3D user interfaces.

## 7.3 General Observations & Discussion

This section outlines some of the other general observations made while undertaking this research, in particular observations made during the user testing phase. Unlike the more specific results outlined above these are more general notes made from observation of the users of the trial 3D systems. This section was intended to be a small section for the minor observations that did not fit into any of the other project results and conclusions, however as the project has developed this section has grown to include many interesting insights from this project including:

- It was initially intended that none of the interfaces or systems developed in this research would "move" the user in space (i.e. leaving movement entirely under his/her control). However it was found that the users do not mind, in fact want (supported by [Mohageg 96]'s previous work), to be able to click items and fly to them, or be smoothly transferred from one region to another. Given the need for this "move the user" capability the builder tools functionality was extended (to enable users of the builder tool to specify actions involving moving the end user in space) and this facility broadened the capacity of the interface application considerably.
- The orientation trial clearly demonstrated that the users did not notice the interface elements unless these elements made themselves obvious (i.e. throbbed, spun or undertook some visual activity to attract the users attention, flat inactive items were readily ignored).
- The most common comment made by users in all trials was that "The 3D system was more enjoyable, but I was more used to the other systems. I think I'd be better with 3D if I had some experience".
- The Flow trial showed that users had an interesting tendency to see items in the top part of the screen before they noticed items in the bottom half of the screen (i.e. this matches with how most people read text which is interesting and perhaps indicates that tool-belts and other move with user tools would be best at the top of the viewing region).
- The orientation of items was less important than initially thought. In many interfaces users were presented with items, images or text that was upside down or on an unusual angle yet they seemed to be able to mentally correct/upright these items. That is not to say that upside down items were preferred, however they did not have the negative impact on the users ability to recognize and select them that was expected.
- Users in general believed they could "click through" partially transparent objects. This indicates that there is a need to either have partially transparent items become solid when the user moves over them (to indicate that you cannot click through) or implement special forms of picking to enable the users to click through these items as they desire.

Overall the results from the user trials of all of the systems have generated very interesting results. With some systems, such as the flow based components (Flow, Tunnel and Circulatory system), showing clear benefits and demonstrating specific real world activities where 3D interfaces can not only be applied, but are more effective than their 2D equivalents. The results obtained in these user trials represent the performances of a group of trial users and as mentioned previously are likely to be relatively representative of the overall population of computer users. However it is important to note that the number of trial users was relatively small and given the variability in some of the results, making statements regarding the relevance of these results for the mainstream population is somewhat difficult. Instead these results are indicators of the likely success of these systems when used in the broader population.

Overall the results have been very successful, with the 3D systems generally regarded as more enjoyable to use and in most cases the 3D interfaces also proved as, or more, effective than their 2D equivalents. The success of the builder tool in providing high level tools that enable inexperienced developers to enter the 3D interface construction ranks, was also very pleasing.

# *Chapter 8: Conclusions*

This research has contributed to the field of 3D interfaces and their development through a range of areas. The positive results for the collection of tools and components developed in this project confirms the contribution to these areas. Within these contributions there are two key advances that represent the biggest steps forward for the field, they are:

***The presentation of bodies of data as an active stream of interactive information. With data propelled through three dimensional space, actively presenting itself to the user as a steady flowing stream of information.***

***The demonstration of innovative specialized high-level development tools and re-usable components, created for the first time for the task of 3D interface development.***

In addition to these main points this research provides a range of conclusions and contributions relating to these two key areas. One key area (described below) incorporates the conclusions relating to the value of high level interface development tools. For the 3D user interface developer such high level tools have not previously been available and as such the construction of 3D user interfaces has always been the domain of experienced and highly skilled graphics programmers only. The simple fact that the number of these potential developers is very low (i.e. highly skilled 3D graphics programmers represent only a tiny proportion of the mainstream computing community) has restricted the development of 3D interfaces in general. One of the key results of this research project was the identification and provision of tools to fill this need for higher level tools that would empower the less skilled computer users and enable them to become developers of interactive 3D user interfaces. The key conclusions reached in this area of research include:

- Eliminating the need for users to understand and utilize programming or scripting systems, to implement interactive elements, is the key to making a high level tool that is effective for all users. This was achieved in the builder tool developed here through an innovative system which enables the user to simply select sensitivities and actions from available sets.
- Providing a set of re-usable “components” each designed and constructed to provide effective methods for implementing common interactive tasks in 3D space enables users to quickly build larger systems (i.e. the building block approach). This is a very powerful mechanism for enabling the development of larger systems as it overcomes one of the major current limitations in 3D user interface development. The limitation is the fact that potential developers do not know which tasks 3D is effective for, nor do they know how or where to locate the tools needed to handle these tasks. By identifying a set of common interactive tasks and developing new reusable components to implement those tasks in 3D, this research project has made it possible for developers to simply use these pre-existing components to construct larger interactive systems.

- High level 3D interface development tools are effective in empowering less skilled developers (as identified by the results of the user trials on the builder tool where even the inexperienced users were able to construct interactive interfaces effectively).
- High level development tools reduce/eliminate errors from the final interactive systems. By working within a constrained system it is much more difficult for users to create interfaces that contain functional errors (of course errors are still possible but tools clearly reduce the rates).
- In summary a high level builder tool takes an extremely complex task and makes it simpler to understand and construct. Even users with no experience in 3D systems or software development can easily create interactive 3D user interfaces. This is achieved by hiding the complex detail, and presenting the user with a simple to use development platform/tool.

As the points above clearly demonstrate the high level "builder tool" developed in this research proved highly effective in its task of providing users (of all skill levels) with a development platform from which complex interactive 3D interfaces can be built, without requiring the developer to have high level programming or 3D skills. In achieving this goal this research project has also identified many areas in which 3D interfaces can not only be effectively applied, but can actually provide more effective interfaces than the existing two dimensional systems. These systems, which were developed to provide a core set of mainstream "components" represent the second key area of contributions and conclusions in this research project. The key conclusions in broad design terms reached in this area of the research include the following (for more specific conclusions on a component by component basis see the next section):

- Three dimensional space offers enormous potential for developing new methods for common interactive tasks.
- Active interfaces that present data to the user rather than wait for the user to interact with them are effective in many mainstream tasks (particularly those incorporating large sets of data).
- Real world experience makes a strong base from which to develop an interactive 3D component (i.e. components based on real world items/tools are generally very effective).
- In 3D environments users need the ability to carry their tools and items with them through the space.
- Three dimensional interfaces are effective (as demonstrated by the results from user trials) in the following common interactive tasks:
    - Storing items for reuse and later retrieval in a local (i.e. with the user) location.
    - Selection of items from small sets
    - Setting values in limited ranges (eg. sizing items)
    - Browsing through large sets of structured data (eg. browsing through movie/music/other libraries).

- Searching through large sets of unstructured data (eg. searching through sets of search results).

The major conclusions reached in this area of the research clearly identify the value of three dimensional user interfaces when applied in real world tasks such as browsing libraries of information or searching through large sets of search results. The fact that today's world is filled with large bodies of information, all at our fingertips through networking facilities such as the Internet/Web and all providing more access to more information than ever before make these new interfaces all the more valuable. The new 3D interfaces developed in this research (which can aid the presentation of such large data sets) offer enormous potential for real world application.

This research has established a range of conclusions of relevance to the field of 3D user interfaces. These conclusions, reached through testing of real-world tasks on real users represent a valuable contribution to the field of 3D user interfaces and open the door to many future innovations in this field. With the introduction of the builder tool developed in this research, the field of 3D user interface development has been opened up to a much broader developer base and it is expected that this contribution alone will enhance and extend the field of 3D user interfaces considerably in the years to come.

# 8.1 Contributions

This research has contributed by making positive steps forward in a range of areas within the field of 3D user interface development. Ranging from outlining new methods and demonstrating specific real world tasks in which 3D interfaces can be effectively applied; to developing new and innovative tools to enable the rapid development of 3D interfaces. The contributions of this research cover a broad area and as the following points outline the contributions represent some significant steps forward for 3D user interfaces.

- The identification of specific real world tasks where 3D can be applied and is effective (eg. browsing large structured sets with the circulatory system and searching large unstructured sets with the flow and tunnel components).
- The design and implementation of new re-usable components to handle these real world tasks.
- The identification of the need for high level development tools to broaden the developer base of capable 3D interface developers.
- The implementation of high-level tools that demonstrably provide this ability to empower inexperienced developers as never before (i.e. through the builder tool).
- The concept and design of a set of components that are capable of functioning when viewed from multiple angles. Thus making them effective when placed at fixed locations in a 3D environment and also making them effective in multi-user environments.
- The introduction of the concept of an active interface that presents itself to the user, enabling the user to passively browse through large sets of information (i.e. flow based systems Flow, Tunnel and Circulatory system).

- The introduction of a flowing 3D steam of data and its effective application in the flow based components (and their real world applications eg. search data, libraries of movies/music/documents/…).
- The design of the new 3D components for setting fundamental values such as the drum and spring-slider. These relatively simple components effectively provide new methods, specific to 3D systems, for providing key pieces of functionality.
- The introduction of the concept of volume based sliders for setting ranging values and how appropriate they are for some specific types of implementations (i.e. the volume based sliders (eg. the Sphere-Slider) represent a new concept in value setting and are very suited to setting values such as ranges, strengths and intensities).
- The concept and implementation of the branching circulatory system (with its automatic flow based traversal system) that enables the users to experience the "flow" style of interface even within structured sets of data.
- The provision of a system to not only create these new "components" but to make them available for re-use in a range of other interfaces. Through the design of the component based model and its association to the high level builder tool.

These contributions provide a collection of key steps forward in the field of 3D user interfaces, with each contributing a small part, yet together representing a significant step forward in 3D user interface research. With the potential to be extended into the future many of the developments outlined here form the basis of a strong future for three dimensional user interfaces.

## 8.2 Future Potential

The potential to extend this research is enormous. The design of the builder tool is based on the concept of extendibility through the creation of new components. This establishes the possibility of creating a broad base of users all developing new components in a range of areas. One of the initial obvious areas is to tap existing 3D systems that have proven effective in specific tasks (eg. Cone trees [Robertson 91]). However there are many possibilities in the type and range of new components that could be developed.

In addition to adding new components there exists the potential to refine many of the components described in this thesis. For example, improving the circulatory system by changing the transitions between rings to be smoothly animated rather than simply viewpoint jumps offers potential; as does the possibility of improving the flow control system. A related area involves the use of audio systems to further improve these flowing systems. In fact, the potential of 3D flowing audio systems represents an area that may offer effective interaction techniques not only for the mainstream user, but also for the blind (eg. flowing "sound items" may be capable of providing blind users with an effective "browsing" interface).

These subtle improvements offer the potential to make these components even more effective in their tasks.

At the broader scale there is enormous potential for the application of 3D components in larger interactive systems (i.e. 3D applications using the range of components to combine into large interactive systems rather than simply components on their own). Taking these re-usable pieces and making larger systems to provide users with valuable computing functionality represents one of the brightest areas of potential. The range of applications that components of this type can be used to implement is enormous, ranging from improving existing interactive systems (eg. the flow interfaces for presenting search data) to entirely new applications to achieve as yet untapped 3D application areas.

Some possible examples include enabling immersive control, through truly 3D control components, in VR systems (i.e. 3D components in a virtual 3D world, eg. 3D tools in a VR surgery application), creating functional applications from within 3D workspaces (eg. browsing applications using move-with-user tools and flow systems), and creating an essentially generic 3D working environment (i.e. quite different in concept to the 2D GUI desktop, as 3D space is not limited and the user can move within the space, this makes the "move-with-user tools" the primary part of this new 3D working environment). As Figure 8.1 demonstrates the 3D components, developed in this research, form the basis of many potential applications, virtual surgery (or real surgery using augmented-reality systems) is just one example of a 3D interactive field into which this research can be extended.

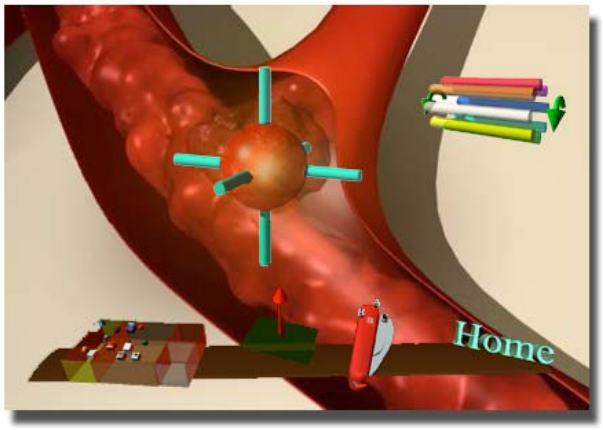



**Figure 8.1: Example component use in VR Surgery concept.**

Some of the additional areas of particular interest include the application of the flowing interfaces in the areas of shopping and education. For the educational example, the ability to enable students to browse through resource materials, choosing their own path presents an interesting area of potential application. In the shopping case the "browsing" capabilities of the flow systems appear to be ideally suited.

Whether these applications are simply the use of the flow components as a viewing mechanism for a real world search tool, or the full integration of 3D interfaces into the desktop of the mainstream operating systems, the future for 3D interfaces and the developers that construct them is very positive.

## 8.3 Final Remarks

This research project has been an enlightening process, starting from a core set of conceptual ideas and eventually developing into the substantive work that has been presented here. The success of this research project has highlighted the potential of 3D graphics when applied in general purpose user interface development. However the full potential of this field is enormous and as yet has barely been touched. Not only does this research offer potential for mainstream computer systems it also extends into related areas such as augmented reality where these 3D user interfaces can be overlaid over the

users real world view. The potential for this field is enormous and the author hopes this research has opened the door for further creative use of 3D graphics to enhance the human computer interface.

# *References*


[Akeley 88] Akeley H, Jermoluk T, *'High Performance Polygon Rendering*', Computer Graphics, 22(4), Proceedings of SIGGRAPH 1988, pg. 239-246.

[Apgar 88] Apgar B, Bersack B, Mammen A, *'A Display System for the Stellar Graphics Supercomputer Model GS1000*', Proceedings of SIGGRAPH 1988, pg. 255-262.

[Beeson 97] Beeson C, '*An Object-Oriented Approach To VRML Development'*, Proceedings of VRML 1997, pg 17-24.

[Bell 01] Bell B, Feiner S and Hollerer T, '*View Management for Virtual and Augmented Reality*', Proceedings of SIGUIST 2001.

[Bell 02] Bell B, Hollerer T and Feiner S, '*An Annotated Situation Awareness Aid for Augmented Reality*', Proceeding of SIGUIST 2002, pg 213-216.

[Bier 93] Bier E, Stone M, Pier K, Buxton W, DeRose T, *'Toolglass and Magic Lenses: The See-through Interface*', Proceedings of SIGGRAPH 1993.

[Bowman 97] Bowman D, Hodges L, *'An Evaluation of Techniques for Grabbing and Manipulating Remote Objects in Immersive Virtual Environments*', Proceedings of 1997 Symposium on Interactive 3D Graphics, pg 35-38

[Brooks 90] Brooks F, Ouh-Young M, Batter J, Kilpatrick P, *'Project GROPE - Haptic Displays for Scientific Visualization*', Proceedings of SIGGRAPH 1990, pg 177-185.

[Bryson 94] Bryson S, '*Tutorial Notes: Introduction to VR*', Proceedings of Virtual Reality and its Applications', CGS, Leeds, UK, 1994.

[Calitz 01] Calitz A, '*Representation of Hierarchical Structures in 3D Space*', Proceedings of AFRIGRAPH 2001, pg 59-64.

[Card 91] Card S, Robertson G, Mackinlay J, *'The Information Visualizer, An Information Workspace*', Proceedings of SIGCHI 1991.

[Chao 01] Chao D, '*Doom as an Interface for Process Management'*, Proceedings of SIGCHI 2001, pg 152-157.

[Cockburn 01] Cockburn A, McKenzie B, '*3D or not 3D? Evaluating the Effect of the Third Dimension in a Document Management System'*, Proceedings of SIGCHI 2001, pg 434-441.

[Cockburn 02] *Cockburn A, McKenzie B, 'Evaluating the Effectiveness of Spatial Memory in 2D and 3D Physical and Virtual Environments'*, Proceedings of SIGCHI 2002, pg 203-210.

[Conner 92] Conner D, Snibbe S, Herndon K, Robbins D, Zeleznik R, vanDam A, *'Three Dimensional Widgets*', SIGGRAPH Symposium on Interactive 3D Graphics 1992, pg 183-188.

[Cosmo 98] SGI Cosmo Worlds Authoring Tool- http://www.sgi.com/software/cosmo/worlds.html

[Cruz 91] Cruz-Neira C, Sandin D, DeFanti T, *'Surround-Screen Projection Based Virtual Reality: The Design and Implementation of the CAVE*', Proceedings of SIGGRAPH 1991.

[Cubaud 01] Cubaud P, Topol A, '*A VRML-based user interface for an online digitalized antiquarian collection*', Proceedings of Web3D 2001, pg 51-59.

[Cubaud 02] Cubaud P, Stowkowski P, Topol A, '*Binding Browsing and Reading Activities in a 3D Digital Library*', Proceedings of JCDL 2002, pg 281-282.

[Damer 96] Damer B, Kekenes C, Huffman T, *'Inhabited Digital Spaces'*, Proceedings of SIGCHI 1996, pg 9-10.

[Darken 93] Darken R, Sibert J, *'A Toolset for Navigation in Virtual Environments'*, Proceedings of SIGUIST 93, pg 157-165.

[Darken 95] Darken R, *'Wayfinding in Large-Scale Virtual Worlds'*, Proceedings of SIGCHI 1995, pg 45-46.

[Deering 95] Deering M, '*HoloSketch: A Virtual Reality Sketching/Animation Tool*', ACM Transactions on Computer-Human Interaction, Vol 2, No 3, September 1995, pg 220-238.

[Diede 88] Diede T, Hagenmaier C, Miranker G, Rubenstein J, Worley W, *'The Titan Graphics SuperComputer Architecture*', Computer, 21(9), 1988, pg 13-30.

[Doom 93] ID Software, 'DOOM - 3D Game', Wikipedia - the Free Encyclopedia, http://en.wikipedia.org/wiki/Doom_computer_game

[Dorner 00] Dorner R, Grimm P, '*Three-dimensional Beans – Creating Web Content Using 3D Components In A 3D Authoring Environment'*, Proceeding of VRML 2000, pg 69-74.

[Eisenberg 97] Eisenberg M, Nishioka A, Schreiner M, *'Helping Users Think in Three Dimensions: Steps toward Incorporating Spatial Cognition in User Modelling*', Proceedings of SIGUIST 1997, pg 113-120.

[Elvins 97] Elvins T, Nadeau D, Kirsh D, *'Worldlets - 3D Thumbnails for Wayfinding in Virtual Environments*', Proceedings of SIGUIST 1997, pg 21-30.

[Fahlen 93] Fahlen L, Brown C, Stahl O, Carlsson C, *'A Space Based Model for User Interaction in Shared Synthetic Environments'*, Proceedings of SIGCHI 1993, pg 43-48.

[Feiner 93] Feiner S, MacIntyre B, Haupt M and Solomon E, *'Windows on the World: 2D Windows for 3D Augmented reality'*, 6th International Symposium on User Interface Software and Technology, pg 145-155, 1993.

[Fisher 86] Fisher S, McGreevy M, Humphries J and Robinett W, *'Virtual Environment Display System'*, Symposium on Interactive 3D Techniques 1986, pg 77-87.

[Fisher 89] Fisher S, *'The AMES Virtual Environment Workstation (VIEW)*', Proceedings of SIGGRAPH 1989, Course Notes, Aug 1989.

[Foley 90] Foley J, vanDam A, Feiner S, Hughes J, '*Computer Graphics: Principles and Practice 2nd Edition*', The Systems Programming Series, Addison Wesley Publishing 1990.

[Foley 92] Foley J, Mitchell C, Walker N, *'Human Computer Interaction Research at Georgia Institute of Technology*', Proceedings of SIGCHI 1992, pg 81-82.

[Fuchs 89] Fuchs H, Poulton J, Eyles J, Greer T, GoldFeather J, Ellsworth D, Molnar S, Turk G, Tebbs B, Israel L, *'Pixel-Planes 5: A Heterogeneous Multiprocessor Graphics System Using Processor-Enhanced Memories'*, Proceeding of SIGGRAPH 1989.

[Graham 95] Graham C, '*Database Visualization and VRML*', Proceedings of VRML '95, pg 21-24.

[Gobbetti 93] Gobbetti E, Balaguer J, *'VB2: An Architecture for Interaction in Synthetic Worlds'*, SIGUIST 93, pg 167-178.

[Haik 02] Haik E, Barker T, Sapsford J, Tranis S, '*Investigation into Effective Navigation in Desktop Virtual Interfaces*' Proceedings of Web3D 2002, pg 59-66.

[Hartman 96] Hartman J, Wernecke J, *'The VRML 2.0 Handbook: Building Moving Worlds on the Web*', Silicon Graphics, Addison Wesley Developers Press 1997.

[Health 03] Better Health Channel - Image Library, http://www.betterhealth.vic.gov.au/bhcv2/bhcsubmit.nsf/imagelibrary?Open, 2003.

[Hinkley 94] Hinkley K, Pausch R, Gable J, Kassell N, *'A Survey of Design Issues in Spatial Input'*, Proceedings of SIGUIST 1994, pg 213-222.

[Hinkley 94] Hinkley K, Pausch R, Goble J, Kassell N, *'Passive real-world interface props for Neurosurgical Visualization*', Proceedings of CHI '94 ACM Conference on Human Factors in Computing Systems 1994.

[Hinkley 97] Hinkley K, Pausch R, Proffin D, *'Attention and Visual Feedback: The Bimanual Frame of reference*', Proceedings of SIGGRAPH 1997, pg 121-126.

[Holm 02] Holm R, Stauder E, Wagner R, '*A Combined Immersive and Desktop Authoring Tool for Virtual Environments*', Johannes Kepler University of Linz 2002.

[Hubona 99] Hubona G, Wheeler P, Shirah G and Brandt M, '*The Relative Contributions of Stereo, Lighting, and Background Scenes in Promoting 3D Depth Visualization'*, ACM Transactions on Computer-Human Interaction, Vol. 6, No. 3, September 1999.

[Iris 96] Silicon Graphics Inc., *'IRIS Universe: Experimental Computing Summer '96'*, 1996.

[Java3D 03] Sun MicroSystems Inc., '*Java3D API'*, http://java.sun.com/products/java-media/3D/

[Johnson 81] Johnson O, Thomas F, *'Disney Animation: The Illusion of Life'*, Abbeville Press Inc 1998.

[Larson 00] Larson K, Dantzich M, Czerwinski M, Roberston G, *'Interactive Posters: Text in 3D: some legibility results'*, CHI '00 extended abstracts on Human factors in computing systems, 2000.

[Leach 97] Leach G, al-Qaimari G, Grieve M, Jinks N and McKay C, '*Elements of a Three-dimensional Graphical User Interface*', Proceedings of the IFIP TC13 International Conference on Human-Computer Interaction 1997.

[Mackinlay 91] Mackinlay J, Robertson G, Card S, *'The Perspective Wall: Detail and Context Smoothly Integrated*', Proceedings of SIGCHI 1991.

[Mereu 96] Mereu S, Kazman R, '*Audio Enhanced 3D Interfaces for Visually Impared Users*', Proceedings of CHI'96 1996.

[Mine 96] Mine M, '*Working in a Virtual World: Interaction techniques used in the Chapel Hill Immersive Modeling program*', University of North Carolina Technical Report.

[Mine 97] Mine M, Brooks F, Sequin C, *'Moving Objects in Space: Exploiting Proprioception In Virtual Environment Interaction*', Proceedings of SIGGRAPH 1997.

[Mohageg 96] Mohageg M, Myers R, Marrin C, Kent J, Mott D, Isaacs P, '*A User Interface for Accessing Content on the World Wide Web'*, Design Briefings - Proceedings of SIGCHI 1996.

[Neider 93] Neider J, Davis T, Woo M, (OpenGL Architecture Review Board), *'OpenGL Programming Guide: The Official Guide to learning OpenGL*', Silicon Graphics, Addison Wesley Publishing 1993

[Parallel 03] ParallelGraphics http://www.parallelgraphics.com - Internet Space Builder™ application and associated development tools 2003.

[Patterson 03a] Patterson D, '*VRML Information: What is VRML and How Can I make VRML Worlds?'*, http://www.scintillatinggraphics.com.au/VRMLmagic/VRMLInfo.html

[Patterson 03b] Patterson D, '*X3D Information: What is X3D and How Can I make X3D Worlds?'*, http://www.scintillatinggraphics.com.au/VRMLmagic/X3DInfo.html

[Pausch 93] Pausch R, Conway M, DeLine R, Gossweiler R, Miale S, *'Alice and DIVER: A Software Architecture for Building Virtual Environments'*, SIGCHI 1993 Conference Companion on Human Factors in Computing Systems, pg 3-4.

[Pausch 95] Pausch R, Burnette T, Brockway D, Weiblen M, *'Navigation and Locomotion in Virtual Worlds via flight into Hand-Held Miniatures'*, Proceedings of SIGGRAPH 1995.

[Perlin 96] Perlin K, Goldberg A, *'Improv: A System for Scripting Interactive Actors in Virtual Worlds'*, Proceedings of SIGGRAPH 96.

[Pierce 97a] Pierce J, Forsberg A, Conway M, Hong S, Zeleznik R and Mine M, *'Image Plane Interaction Techniques in 3D Immersive Environments'*, 1997 Symposium on Interactive 3D Graphics, pg 39-44.

[Pierce 97] Pierce J, Audia S, Burnette T, Christiansen K, Cosgrove D, Conway M, Hinkley K, Monkaitis K, Patten J, Shochet J, Staack D, Stearns B, Sturgill C, Williams G, Pausch R, *'Alice: Easy to Use Interactive 3D Graphics'*, Proceedings of SIGUIST 1997, pg 77-78.

[Pierce 99a] Pierce J, Steams B, Pausch R, *'Voodoo Dolls: Seemless Interaction at Multiple Scales in Virtual Environments'*, 1999 Symposium on Interactive 3D Graphics, pg 141-145.

[Pierce 99] Pierce, J., Conway, M., van Dantzich, M., and Robertson, G., '*Toolspaces and glances: storing, accessing, and retrieving objects in 3D desktop applications'*, Proceedings of Symposium on Interactive 3D Graphics, April 1999, pp. 163-168.

[Pimentel 94] Pimentel K, Teixeira K, *'Virtual Reality: Through the new Looking Glass' 2nd Edition*, Intel/McGraw-Hill 1995.

[Racer 00] LucasArts Inc., 'Star Wars™ - Episode 1 Pod-Racer', http://www.lucasarts.com/products/starwarsracer/splash.htm

[Robertson 91] Robertson G, Macinlay J, Card S, *'Information Visualization using 3D Interactive Animation'*, Proceedings of SIGCHI 1991, pg 461-462.

[Robertson 91b] Robertson G, Mackinlay J, Card S, *'Cone Trees: Animated 3D Visualizations of Hierarchical Information*', Proceedings of SIGCHI 1991.

[Robertson 98] Robertson G, Czerwinski M, Larson K, Robbins D, Thiel D and Dantzich M, '*Data Mountain: Using Spatial Memory for Document Management'*, Proceedings of SIGUIST 1998, pg 153-162.

[Robertson 00] Robertson G, Dantzich M, Robbins D, Czerwinski M, Hinkley K, Risden K, Thiel D, Gorokhovsky V, '*The Task Gallery: A 3D Window Manager'*, Proceeding of SIGCHI 2000, pg 494-501.

[Runyon 87] Runyon S, *'AT&T Goes to 'Warp Speed' with its Graphics Engine'*, Electronics Magazine, July 1987.

[Salzman 96] Salzman M, Loftin R, *'ScienceSpace: Lessons for Designing Immersive Virtual Realities'*, Proceedings of SIGCHI 1996, pg 89-90.

[Schonhage 00] Schonhage B, van Ballegooij A, Eliens A, '*3D Gadgets for Business Process Visualization —a case study—',* Proceedings of VRML 2000, pg 131-138.

[Shaw 92] Shaw C, Liang J, Green M, Sun Y, *'The Decoupled Simulation Model for Virtual Reality Systems'*, Proceedings of SIGCHI 1992, pg 321-328.

[Snibbe 92] Snibbe S, Herndon K, Robbins D, Conner D, vanDam A, *'Using Deformations to Explore 3D Widget Design'*, Proceedings of SIGGRAPH 1992, pg 351-352.

[Stevens 94] Stevens M, Zeleznik R, Hughes J, *'An Architecture for an Extensible 3D Interface Toolkit'*, Proceedings of SIGUIST 94, pg 59-67.

[Stoakley 95] Stoakley R, Conway M, Pausch R, *'Virtual Reality on a WIM: Interactive Worlds in Miniature*', Proceedings of CHI 1995, pg 265-272.

[Strauss 93] Strauss P, *'IRIS Inventor - A 3D graphics toolkit'*, ACM Proceedings of the 8th Annual Conference on Object Oriented Programming 1993, pg 192-200.

[Streiner 96] Streiner D, *'Maintaining Standards: Differences between the Standard Deviation and Standard Error, and When to Use Each'*, Canadian Journal of Psychiatry, pg 498-502, 1996.

[Sutherland 65] Sutherland I, *'The Ultimate Display'*, Proceedings of IFIP '65, 2, pg 506-508, 582-582.

[Sutherland 68] Sutherland I, *'Head Mounted Three Dimensional Display'*, AFIPS Fall Joint Computer Conference '68, pg 757-764.

[Tanriverdi 01] Tanriverdi V, Jacob R, '*VRID: A Design Model and Methodology for Developing Virtual Reality Interfaces*', Proceeding of VRST'01 2001, pg 175-182.

[vanDam 88] vanDam A, *'PHIGS+ Functional Description, Revision 3.0'*, Computer Graphics, 22(3), July 1988.

[Viega 96] Viega J, Conway M, Williams G, Pausch R, '*3D Magic Lenses*', Proceedings of SIGUIST 1996, pg 51-58.

[Viire 97] Viire E, '*Health and Safety Issues for VR'*, Communications of the ACM, Vol. 40, No. 8, August 1997, pg 40-41.

[Vince 95] Vince J, '*Virtual Reality Systems*', ACM Press 1995.

[Wakita 03] Wakita A, Matsumoto F, "*Information Visualization with Web3D*", Computer Graphics, 37(3), 2003, pg 29-33.

[Wang 99] Wang Y, MacKenzie C, '*Object Manipulation in Virtual Environments: Relative Size Matters*', Proceeding of SIGCHI 1999, pg 48-55.

[Ware 97] Ware C, Fleet D, *'Context Sensitive Flying Interface'*, ACM Symposium on Interactive 3D Graphics 1997, pg 127-130.

[Waterworth 94] Waterworth J, Serra L, *'VR Management Tools: Beyond Spatial Presence'*, Proceedings of SIGCHI 1994, pg 319-320.

[Web3D 03] Web3D Consortium, '*Information technology — Computer graphics and image processing — Extensible 3D (X3D)*', http://www.web3d.org/specifications/ISO-IEC-19775/index.html

[Wloka 95] Wloka M, Greenfield E, *'The Virtual Tricorder: A Uniform Interface for Virual Reality*', Proceedings of SIGUIST 1995, pg 39-40.

[Wolfenstein 92] ID Software, 'Wolfenstein 3D', Wikipedia - the Free Encyclopedia, http://en2.wikipedia.org/wiki/Wolfenstein_3D

[Xiaoguang 99] Xiaoguang Z, Dongmu W, Bingrong H, '*An Object-Oriented Data Framework for Virtual Environments with Hierarchical Modeling'*, ACM SIGSOFT Software Engineering Notes vol 24 No 1, pg 65-68.

[Zeleznik 93] Zeleznik R, Herndon K, Robbins D, Huang N, Meyer T, Parker N, Hughes J, *'An Interactive 3D Toolkit for Constructing 3D Widgets'*, Proceedings of SIGGRAPH 1993.

[Zhai 94] Zhai S, Buxton W, Milgram P, *'The Silk Cursor: Investigating Transparency for 3D Target Acquisition'*, Proceedings of SIGCHI 1994, pg 459-464.

[Zhai 96] Zhai S, Buxton W, Milgram P, *'The Partial Occlusion Effect: Utilizing Semitransparency in 3D Human-Computer Interaction'*, ACM Transactions on Human-Computer Interaction Vol 3 No. 3 September 1996, pg 254-284.

[Zimmerman 03] Zimmerman G, Barnes J and Leventhal L, '*A Comparison of the Usability and Effectiveness of Web-Based Delivery of Instructions for Inherently-3D Construction Tasks on Handheld and Desktop Computers*', Proceedings of Web3D 2002, pg 49-54.

# *Appendix A: User Testing Configuration*

This section outlines the methods and systems used for the human testing elements of the 3D SPACE project. The user testing was undertaken based on the following principles:

- All user testing is voluntary and users may stop at any time.
- All users are informed of trial contents prior to undertaking test.
- No part of the trials is intended to mislead the users (i.e. none of the trials involve deception of the user in any form).
- No personal or private user data is requested or recorded, only responses to questions relating to system performance are recorded.
- All users are tested with a single consistent computer system (i.e. eliminate system performance as a variable).
- Use of a mainstream computer system for all user testing (i.e. users are not required to learn any new hardware systems or devices).
- Limit Trial duration to 40 minutes (i.e. to ensure all users results are not effected by fatigue)
- All users are offered the opportunity to be informed of the final findings of the research.

As the intention of the project was to create new 3D systems for real-world functional tasks, the testing system used was not a specially enhanced 3D system, instead it was a relatively mainstream system, one that would be commonly found on desktops around the world. Although it would have been possible to use high end 3D systems and interaction devices, the focus of this research is on the content of the worlds rather than the specialist interaction devices and systems in which they could be applied. Perhaps using more advanced hardware and devices (eg. immersive VR type environments) represents an area of future potential for this research.

## The Hardware

The focus of the user trials was the comparison between the 3D interfaces and as such it was critical to eliminate all other varying factors (including system hardware performance) that could influence user responses. As a result the option of testing the interfaces with a large group of users, each with potentially varying systems (i.e. by using the WWW to trial the system on many users) was considered. It was in fact even trialed in the early stages, but in the end not used. The inconsistencies in system performance (of such a WWW based system) introduced too many variables into the tests. Instead a single testing platform (as described below) was developed and used for all trials. By using a single consistent system for all trials we eliminated the risk of system variations effecting the results obtained.

This system has the following hardware features:

- Main System (Apple Powerbook G4)
  - Processor: 1GHz PowerPC G4
  - Memory: 512MB PC133 SDRAM
  - Disk: 60GB Ultra ATA/66
- Graphics System
  - Graphics: ATI Mobility Radeon 9000 with 64MB of DDR SDRAM
  - Display: 15.2-inch (diagonal), 1280x854 resolution, TFT
- Input Devices
  - Trackpad or
  - Microsoft Wireless Optical Mouse

### *Why a Laptop?*

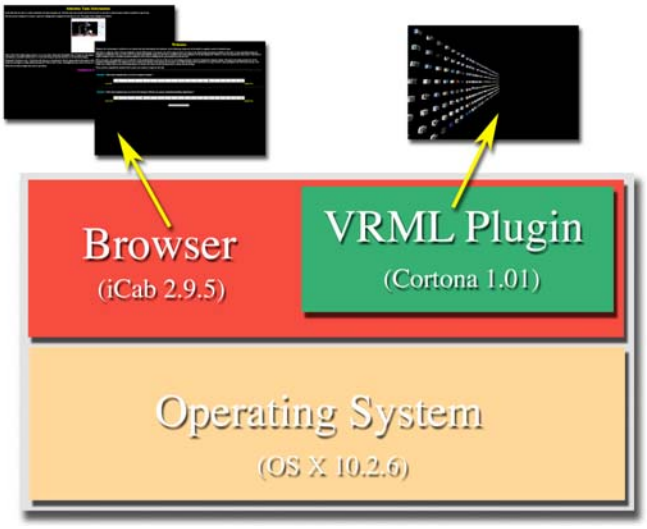
The testing system was implemented as a portable machine (i.e. a laptop) primarily because it was far more convenient for user testing. As a portable system it could be taken to various locations to enable more users to undertake the trials. The performance of the system was suitable for the trials being undertaken (with interactive display/frame rates never falling below 10fps) and the portability enabled the trials to be moved to a range of locations, enabling access to a greater variety of users. As a testing platform it was ideally suited for its task.

### *Impact of a Mouse / Trackpad?*

To maintain the mainstream usability of the system the input devices used were both two degree of freedom in function (X and Y axes only). This limits the users ability to move the pointer into the depths of the screen. Although limiting in one way this makes use of the most widely available form of interaction device and as such means that the demonstrated effectiveness (or otherwise) of the trial systems applies on the bulk of current computers. Six degree of freedom devices could add additional functionality and this is an area for future research.

## The Software

Each trial is made up of a series of local web pages (i.e. no network or download delays), starting with simple informative HTML pages and then moving on to the interactive VRML worlds that make up the trial 3D interfaces. As these pages are all in standard formats (eg. HTML, VRML) it would be possible to run these trials on a range of systems, the software system used as the base for our trials is as follows:




As displayed in the previous diagram the Operating System is OS X Version 10.2.6. On top of the operating system we are running the browser application iCab 2.9.5 (OS X) in kiosk mode (i.e. full screen without any menu bars or window borders). Within the iCab browser application we are using the Cortona (Version 1.01d13) VRML plug-in to display the interactive VRML content (again in full screen without menu bars or console windows). The key features (for the purpose of user testing and result collection) this software system supplies are:

- Browser capable of displaying in full-screen (windowless mode)
- Browser capable of storing cookies (used for data recording)
- Browser capable of using suitable VRML plug-in
- VRML plug-in capable of supporting the following VRML features:
  - VRML 97 standard support including (sensors, routes and script nodes)
  - VRML JavaScript support

# The Testing Process

All user tests run for a period of approximately 30 minutes (although there are no specific time constraints[33]) and follow the outline given below.

- Participant volunteers to be part of trials.
- Participant reads the *Explanatory Statement for Research Participants*[34].
- Participant signs the *Consent Form*[35].

---

[33] Each trial is designed to take less than 40 minutes, even for the most inexperienced of users.

[34] See Appendix B for this document.

- Participant goes to the computer and begins the trial by following the on-screen instructions.
  - Initial instructions inform participant of the task he/she is required to complete.
  - Interactive 3D task then begins.
  - Task concludes and participant is presented with a questionnaire about the task.
  - Questionnaire is completed and participant is transferred to the "Initial instructions" for the next task.
- Participant completes all tasks and questionnaires in the trial and receives thanks for his/her contribution.

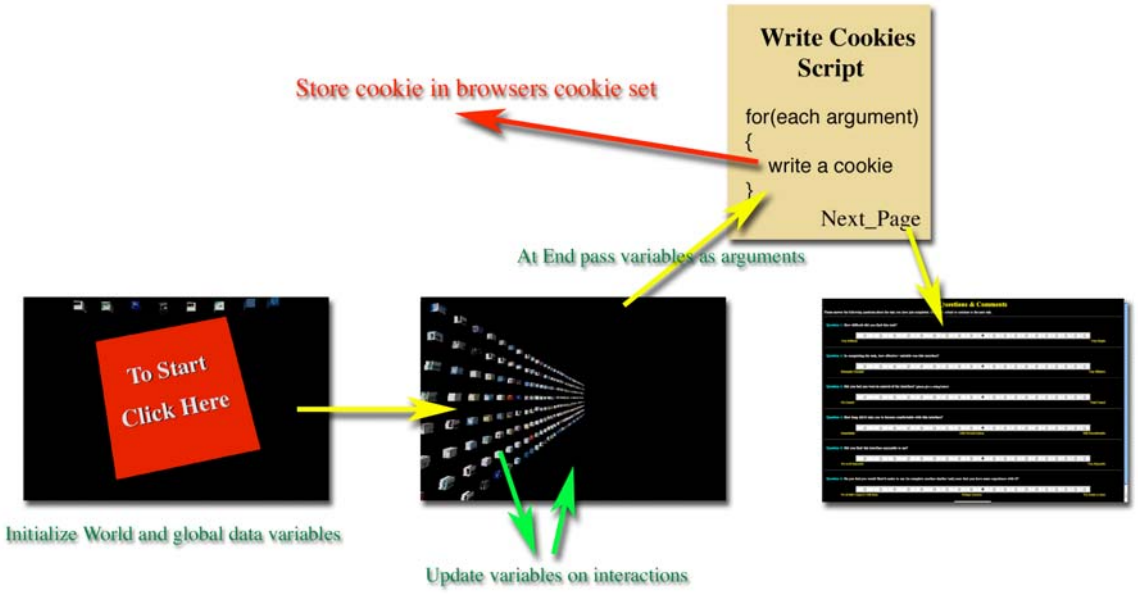
# Data Recording

The user trials provide critical feedback, both qualitative and quantitative, on the interactive systems being tested. All of this data is stored in the form of cookies within the browser application. The systems that were used to record the relevant data are described here.

## *Recording the Quantitative Data*

Recording quantitative results (which is done in the background of the trial task, without the users being aware of it and includes the recording of values such as time taken to complete tasks, number of clicks etc) involves several steps and makes use of a number of software features. The system used to record quantitative results is outlined below.

[35] See Appendix B for this document.




The diagram above demonstrates the overall recording system and how it operates. The system begins with the user loading the VRML world (i.e. the 3D trial system). This initialises the values within the world including the global VRML nodes specifically created for storing the numeric quantitative values. The user then begins the trial and begins interacting with the VRML world. During this interaction the scripts in the VRML world record the users actions (eg. every time he/she clicks on an incorrect target the script for that target adds one to the global numeric *incorrect_clicks* variable). At the completion of the trial the VRML world transfers to the *write_cookies* script passing as arguments all of the relevant data. In addition to this main recorded data the VRML worlds close script also adds an extra value which is the URL of the page to be transferred to after the cookie writing is complete (eg. Next_Page=QuestionnaireN.html). The *write_cookies* script then takes all of the arguments passed in and writes each argument out as a cookie value in the browser. When the script finds the Next_Page argument it gets the URL and transfers to that page, thus completing the recording of the quantitative data for the trial.

This system works well in that the *write_cookies* script is consistent across all trials and it is simply a matter of passing to it with the relevant results on a trial by trial basis. In addition to the *write_cookies* script (which takes the arguments and writes out the cookies) there is also a *view_cookies* script which reads the cookies and displays the results they represent in a more accessible form[36].

[36] This is generally used at the end of the final trial to allow easy access to the data.

## *Recording the Qualitative Data*

Recording qualitative results (in the form of responses to online questionnaires) also involves several steps and makes use of a number of software features. The system used to record qualitative results is outlined below.

The diagram above demonstrates this recording system and how it operates. The system begins with the user loading the questionnaire web page. This page is primarily a large form into which the user enters responses to the questions (in addition to the user entered fields the form also includes a hidden field containing the Next_Page URL argument). When complete the user submits the form and this action passes the form data to the *write_cookies* script. The *write_cookies* script then takes all of the arguments passed in and writes each argument out as a cookie value in the browser. When the script finds the Next_Page argument it gets the URL and transfers to that page, thus completing the recording of the qualitative data for this trial.

# *Appendix B: User Testing Documents*

## Explanatory Statement for Research Participants

**Date:** Monday, August 12, 2002

**Project Title:** 3D SPACE (3D Special Project in Advanced Computer Environments)

**Project Number**: R0226

Mr. Dale Patterson is doing research under the supervision of Dr. Michael Rees, an Associate Professor in the Department of Information Technology towards a Doctor of Philosophy (Ph.D.) at Bond University. The aim of this research is to assess the viability of 3D interaction for many common computer tasks.

The testing cycle is part of a larger project whose aim is to develop new and innovative three-dimensional human-computer interfaces. It is through trials on users such as yourself that we are able to determine which techniques are the most effective.

The test you are about to undertake takes approximately 30 minutes to complete and involves interaction with various three-dimensional computer interfaces. There is a very slight risk that some participants may experience a sense of disorientation during testing. Feel free to stop at any time should this occur. As a trial user you will be asked to undertake a series of simple tasks (such as selecting particular items from within a set) in the example 3D environment. The system will record how efficiently you complete those tasks and from this data we can determine which techniques are the most/least effective. These tests are completely anonymous and at no point will you be asked to provide any personal or private information. The primary intent is to clarify which interface techniques are good and which are bad. As a result you will find some interfaces to be difficult to use. This is not a fault on your part but an integral part of the test itself. As a test user you should simply attempt to complete the specified tasks to the best of your ability. If you find one of the trial systems difficult to use simply give it a poor rating when its test is complete.

The information we record primarily measures how successful you are in completing the requested tasks, this information is purely used to assess the performance of the interface. No records are kept relating individual participants to performance scores.

Although the test is estimated to take 30 minutes there are no strict time requirements, so feel free to take your time to complete the test. Should you have any difficulties during the testing procedure feel free to stop the test at any time (One of the above mentioned researchers will be available at all times during the testing procedure. If you have any concerns/questions during testing feel free to consult them). These tests are entirely voluntary and any time you contribute will be appreciated.

We hope through this research to demonstrate some of the obvious advantages of using an extra dimension in the design of the human-computer interface. And in the broader view we hope this research will identify methods of making computers easier and more effective to use.

If you have any queries or would like to be informed of the aggregate research finding, please contact:

Name of Student Investigator: Mr. Dale Patterson Signature: _____

Name of Principal Investigator/Supervisor: Dr. Michael Rees Signature_____


Address: School of Information Technology, Bond University 4229.

Telephone: (07) 5595 3351

Fax: (07) 5595 3320

Email: michael_rees@bond.edu.au

Should you have any complaint concerning the manner in which this research is conducted, please do not hesitate to contact Bond University Research Ethics Committee, quoting the Project Number (above):

The Complaints Officer

Bond University Human Research Ethics Committee

Bond University

Gold Coast, 4229.

Telephone (07) 5595 4001 Fax (07) 5595 1747

E-Mail Jodie Maguire jodie_maguire@bond.edu.au

# Participant Informed Consent

I agree to take part in the above Bond University research project. I have read the above Explanatory Statement. I am willing to:

- use a computer to undertake a series of simple interactive tasks in a three-dimensional user interface.
- complete questionnaires asking me about the three-dimensional user interfaces with which I have been interacting

I understand that any information I provide is anonymous, and that no information that could lead to the identification of any individual will be disclosed in any reports on the project, or to any other party.

I also understand that my participation is voluntary, that I can choose not to participate in part or all of the project, and that I can withdraw freely at any stage of the project.

Name (please print): .....

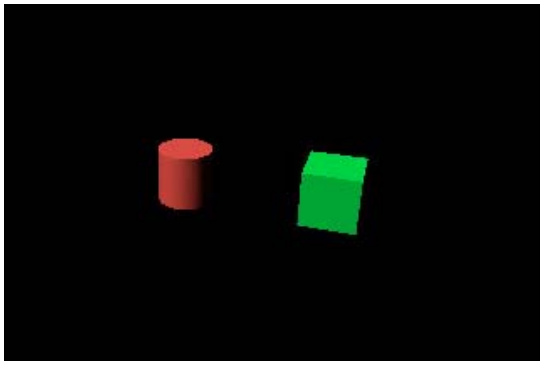
Signature: .....

Date: .....

# 3D Interface Development Information

## *The Task:*

The task involves constructing a simple pair of interactive items as shown below. The items each have a unique colour and a unique associated action. That is:



- The Green box object on the right will spin in place when the mouse moves over it and when clicked on it will link to the URL: www.bond.edu.au
- The Red cylinder on the left will fade in and out when the mouse moves over it and when clicked it will link to the URL: www.bond.edu.au/it/

## *Information on using the Builder Tool:*

The following notes briefly describe the key functionality available through the builder tool, including how to…

- **Add Items (like objects and lights) to the Interface**:
  - Use the Menu *Objects* - *Add Object* to bring up dialog to select type of object to add.
  - Use the Menu *Lights* - *Add Light* to bring up dialog to select type of light to add.
  - Use the Menu *Cameras* - *Add Camera* to add a new (default) camera.
- **Positioning Items in the Space**:
  - Using the mouse simply click on the item you wish to move/rotate, this will highlight (i.e. bright green outline will appear).
  - Then drag (i.e. click down and drag while button down) to move the item in the space.
  - Note that the RIGHT button will ROTATE the item in place where the LEFT button will MOVE the items location.
  - When you have the location/orientation as desired simply release the button.
- **Set the Attributes (like colour and interactivity) of Items in the Space**:
  - Using the mouse simply click on the item you wish to move/rotate, this will highlight (i.e. bright green outline will appear).
  - With item still selected use the menu *Objects* - *Edit Attributes…* to bring up attributes dialog.
  - In this dialog use the buttons to set attributes (eg. set colour by clicking the colour swatch, texture by clicking the texture swatch or an action by clicking relevant button eg. Touch).
- **Saving & Previewing the Final Result**:
  - To Save - Use the Menu *File* - *Save As…* (and specify a filename/location - NOTE: simply save as “Trial” in the current directory for this test).
  - To Preview - Use the menu *File* - *Preview in Explorer*.

Should you have any other questions or queries please ask the trial coordinator for more information.

# *Appendix C: Ideas for Designing Components*

This section covers the design of the components from the perspective of the developer/designer and looks at issues such as how to come up with interface ideas and how to assess which are appropriate. During the design phase of the research it was found that creating new concepts and solving the design issues became easier with experience. This section is intended to outline some of the methods that were used in creating the new interface designs.

In the design stage the primary objective is to come up with ideas for how each of the various interactions could be implemented in a three dimensional environment. This is where the creative side of this research comes to the fore. The opportunity to design components, and concepts, led to many creative ideas, the bulk of which never got past the drawing board. However from this mass of possibilities came the set of likely candidates (many of which are described in earlier sections as final systems).

Although each design is different and inspiration can come from a range of areas. The general design process can be summarized by the following series of steps:

**STEP 1: Look at the task. Think it through. Be creative with possibilities.**

Get as much information as possible on the task and be as creative as possible with solutions. At this stage there is no need to consider implementation issues or other restrictions. Simply be creative with possible options. The activities to undertake at this stage include:

- ***Look at the Task to be Handled***

  Take the time to look at the task itself, not simply from its application in the computer system but as a general activity. What is this task really about. Try to describe it in a single clear statement.

- ***Try to find a real world example of this task***

  If appropriate, finding a real world example of a task (or something similar) can be enormously beneficial. If you can find something that you can go and actually do, then this simple activity can often expose the core details of the task in a way that is otherwise very difficult to achieve.

- ***Look at existing methods for handling this task***

  This is a critical step for several reasons. Firstly there is no point in reinventing the wheel, so if a solution already exists then make sure you find it. Secondly there may be existing systems that solve similar (but not identical) problems. You can learn an enormous amount from such systems. For example in the case of 3D interfaces there are often 2D interface components for handling the exact task you are looking at solving in 3D space. Although the 2D components cannot be applied directly into 3D, there may be principles that you can learn or apply in your own designs.

- ***Can this task be broken down***

  In some cases a larger task can be broken into a set of smaller tasks (each of which can be handled by an independent component). So a set of smaller components (even if those components need to be developed) may be able to be put together to solve this problem.

- ***Identify if this task is suited to a 3D Component***

  Some items and tasks are unsuited to 3D representation. For example textual information is the two dimensional method of presenting our verbal language.

  As a result presenting a 2D textual page or image in 3D is most effectively achieved by leaving the textual content in its 2D form (i.e. text on a plane in 3D space). You could look at novel ideas like using an audio presentation of text, but generally speaking as text is inherently 2D you are better off leaving it as a 2D item and focusing on the tasks for which 3D solutions are more effective.

- ***Try to find real world solutions for this task***

  In most cases the best user interfaces are based on real world solutions that can be applied in a computer interface (e.g. the GUI menu is based on the menu we see at restaurants). This technique of making use of "real world" experience is very effective. The "real world" solution does not necessarily need to apply to a similar real world task, in fact it may be totally unrelated. The advantage of modeling interface designs on real world systems is that the user may have an inherent understanding of how these "real" systems work and as such the interface may be more effective.

- ***How would I do it?***

  Sometimes when designing solutions to interface design problems you can get caught not thinking of the problem from the users perspective. The best way to avoid this is to simply work out what would be fastest and easiest for you personally if you were undertaking this task.

- ***Apply strengths of 3D (i.e. depth, motion etc) to this task?***

  Knowing the key areas in which a 3D interface has advantages, do any of these areas lead themselves to use in this task?

  This technique needs to be applied with caution as it is easy to try to funnel all your interaction tasks into solutions that make use of the best 3D features. This is NOT a good idea. It is good to try to apply the best of 3D to a problem, but it is important not to reshape the problem to fit the best of 3D.

**STEP 2: Convert the idea into a physical design.**

At this point you'll have a basic concept in mind for the interface. Generally you'll need to refine the concept considerably before you're ready to start building it. This refining (which includes taking it from a conceptual idea into a set of more physical pieces) can incorporate many steps including:

- ***Draw it***

  Draw/Visualize the component. Sometimes this is harder than it sounds and often it is not until you draw it that you can see the practical challenges and issues it presents (e.g. items get in each others way, particularly items that are close to the viewers location)

- ***Describe it***

  Can it be described concisely? If you can't describe it in a quick sentence then it needs more work. This is particularly true when dealing with interactive elements, as its easy to draw a design which looks right under one circumstance but when the activity takes place it becomes unsuitable. Describing the activity and the effect it has is critical.

- ***Does it work in this form or are additions needed?***

  Sometimes you have a conceptual idea which seems ideal for dealing with the key feature of its task. However you still need to provide a method for handling the additional elements that make the system work (e.g. aside from the main flow of items the Flow component also needs a mechanism to control the flows speed etc (but how will they work?). Make sure you describe the complete solution before moving on.

- ***Does it have unnecessary extras***

  The key here is to follow the principle of KEEP IT SIMPLE. The orientation aids trial clearly showed how the simplest system is usually the best. In that example, despite the graphical richness and extra information in some of the systems, the users preferred the simple one (i.e. more information is not always better). So keep it simple and do not over-solve the problem.

**STEP 3: Plan to build it**

This is the final stage of the design process and involves taking the conceptual design from above and working out how to implement it as an actual system. This requires a more detailed look at implementation issues and limiting factors.

- ***Implementation Limiting Factors***

  Whether it be a performance limitation or another type of factor, in all cases the final system cannot work if the implementation cannot be built. Some factors can be bypassed or worked around and this is worth trying, however be aware of spending more time on the workaround than it would take to fix the original problem. An example of this type of problem is the fact that partially transparent items can't be clicked through in VRML. This means that despite the fact you can design an interactive see/click through component you can't build it using current VRML technology.

- ***Can it be built?***

  Taking into account limiting factors, is it possible to implement this design? If so what pieces will be required (e.g. you my need to create special

geometric structures or perhaps specialized scripts/programs to supply the needed actions).

- ***What are the variations that could apply to this and should I look at them***

  Most of the designs are flexible to a range of subtle changes, some of which may make large differences in performance. As trial systems it is often not worth implementing large numbers of variations (i.e. best to focus on the main interface and demonstrating its usefulness). However assessing the range of variations with the objective of locating any versions which are of particular potential is worthwhile. This was clearly demonstrated by the orientation aids trial. In that trial the system preferred by users was actually the final variation system added to the trials.

- ***How to build it***

  Prepare pieces to make final item easier to build (e.g. most components require specialized geometry and scripts. Work out their functionality and build them first.)

The design at this stage has undergone a range of refinements and simple assessments such that it now represents a system that should provide a worthwhile user trial.